Dependency Graph-Based Statistical Machine Translation

Liangyou Li

B.Sc., M.Sc.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Computing

Supervisors: Prof. Qun Liu and Prof. Andy Way

December 2016

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

(Candidate) ID No.:

Date:

Acknowledgments

I would first like to express my utmost gratitude to my fantastic supervisors Prof. Qun Liu and Prof. Andy Way for their guidance and encouragement throughout my Ph.D. study. Their profound professional knowledge, sharp insight and enormous support not only keep my research on the right track but also help to build solid foundations for my future career. Their guidance, from general research directions to concrete language and scientific writing, helps me build up ability and confidence to carry out my research and future work. I am also very grateful to examiners Prof. David Chiang and Dr. Jennifer Foster for their insightful feedback and creative comments which make the thesis more solid. I should also thank Dr. Alistair Sutherland for chairing my viva and thank Dr. Jennifer Foster and Dr. John McKenna for providing insightful suggestions on my transfer report.

I thank my excellent colleagues and friends who build a great research atmosphere around me and provide a great help in my life. Particular thanks go to Dr. Junhui Li, Dr. Jun Xie and Dr. Zhaopeng Tu for their help on the opportunity of being in DCU, Dr. Jun Xie again for his help on the first research work of the thesis, Dr. Xiaofeng Wu, Dr. Qi Zhang, Dr. Xiaojun Zhang, Dr. Jinhua Du, Ning Zhang, Jian Zhang, Longyue Wang and Tengqi Ye for their generous help in my study, life, and career. I thank Chris Hokamp for organizing a machine learning group where I learned a lot, Dr. Joachim Wagner for providing technical support, and Piyush Arora for actively organizing activities. I must also thank ADAPT/CNGL management team for offering all the fantastic facilities and support for my study.

Thank the Marie Curie scheme for providing generous research funding to me to allow me to gain research experiences abroad. I thank the EXPERT project which provided innovative research and training to me in the field of machine translation technologies. I am also very fortunate to have had chances to visit the MT group at the University of Amsterdam and Hermes Traducciones in Spain where I met a bunch of excellent people. Thank them all. I would also like to thank Science Foundation Ireland for their generous funding which allows me continuing my research.

Last but certainly not the least, I extend my deepest gratitude to my family. They have always been being there on my side and supporting me for everything in my life. It is their encouragement and generosity that help me go through the ups and downs in the course of my education.

Contents

List of Figures

List of Tables

Abbreviations

CCG Combinatory Categorial Grammar.

DBG Dependency-Bigram Graph.

DE–EN German–English.

DEG Dependency-Edge Graph.

Dep2Str Dependency Tree-to-String.

DSG Dependency-Sibling Graph.

ERG Edge Replacement Grammar.

HD Head-Dependent.

HPB Hierarchical Phrase-Based.

MDD Mean Dependency Distance.

MT Machine Translation.

NLP Natural Language Processing.

NRG Node Replacement Grammar.

POS Part-of-Speech.

SCFG Synchronous Context Free Grammar.

SERG Synchronous Edge Replacement Grammar.

SHRG Synchronous Hyperedge Replacement Grammar.

SMT Statistical Machine Translation.

SNRG Synchronous Node Replacement Grammar.

STSG Synchronous Tree Substitution Grammar.

ZH–EN Chinese–English.

Abstract

Dependency-Based Statistical Machine Translation

Liangyou Li

Statistical Machine Translation has been shown to benefit from complex linguistic structures. However, previous work mainly focuses on sequences and trees. In this thesis, we build dependency graphs which are constructed from dependency trees and uniformly represent both dependency relations and sequential relations, including bigram relations and sibling relations. We propose translation models to translate these graphs into target strings and conduct experiments on Chinese→English and German→English translation tasks.

As a motivation, we firstly present a pseudo forest-to-string model which improves a dependency tree-to-string model by dependency decomposition. The decomposition takes sibling relations into consideration which results in more rules being used and thus a higher phrase coverage. Experiments show that such decomposition is beneficial to translation performance. Integrating phrasal rules further improves our model.

Then, we propose a segmentational graph-based translation model. It segments graphs into subgraphs and generates translations from left to right by combining translations of these subgraphs. The graphs explicitly combine dependency relations and bigram relations. In experiments, the graph-based model outperforms both the phrase-based model and treelet-based model. In addition, we improve this model by using a graph segmentation model to take source context into consideration.

Furthermore, inspired by using tree grammars to translate trees, we propose recursive graph-based translation models by using graph grammars. An edge replacement grammar is used to translate dependency-edge graphs which are converted from dependency trees by labeling edges to naturally take sibling relations into consideration. A node replacement grammar is used to translate dependency-sibling graphs which explicitly add sibling links to dependency trees. Experiments show that our models are significantly better than the hierarchical phrase-based model.

When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

— Warren Weaver

Chapter 1

Introduction

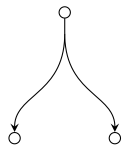
The task of Machine Translation (MT) is to translate texts from one language to another language. It is one of the most important applications of Natural Language Processing (NLP). Along with the availability of large corpora, Statistical Machine Translation (SMT) has developed rapidly in the last decades. SMT is formulated as follows: given a sentence s in a source language, find a sentence t in a target language which has the highest probability of being the correct translation of s according to the distribution $p(t|s)$.

1.1 Definitions

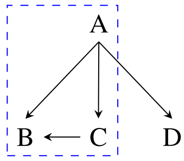
Before reviewing SMT model, we first present formal definitions of several fundamental structures which will be used throughout the thesis. Note that all of the following structures are directed (ordered) and connected. This means that for any two nodes in a structure, there is at least one path (without considering edge directions).

Definition 1.1. A *labeled* and *ordered hypergraph* is a tuple $\langle V, E, \phi, \psi \rangle$, where

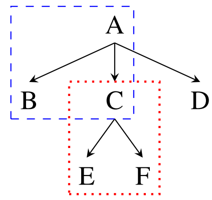
- V is a finite set of nodes.
- $E \subseteq V^+$ is a finite set of edges.



(a) Hyperedge. Node and edge labels are ignored.



(b) Graph. Edge labels are ignored.



(c) Tree rooted at A. Edge labels are ignored.

Figure 1.1: Illustration of various structures mentioned in this thesis. (a): an example of a hyperedge connecting three nodes. (b): *ABC* in the dashed rectangle is a subgraph. (c): *ABC* in the dashed rectangle is a treelet while *CEF* in the dotted rectangle is a subtree. *CEF* is a treelet as well while *ABC* is not a subtree.

- $\phi : E \to C$ is a function which assigns a label from C to each edge.
- $\psi : V \to D$ is a function which assigns a label from D to each node.

In a hypergraph, an edge which connects three or more nodes is called a *hyperedge*, as illustrated in Figure 1.1a. A graph is a special case of a hypergraph, where each edge connects two nodes as defined in Definition 1.2.

Definition 1.2. A *labeled* and *ordered graph* is a tuple $\langle V, E, \phi, \psi \rangle$, where

- V is a finite set of nodes.
- $E \subseteq V^2$ is a finite set of edges.
- $\phi : E \to C$ is a function which assigns a label from C to each edge.
- $\psi : V \to D$ is a function which assigns a label from D to each node.

In this thesis, all graphs are connected. Figure 1.1b shows a graph example. Although in Definition 1.2 a graph is both node-labeled and edge-labeled, in this thesis for simplicity a graph is either node-labeled or edge-labeled by default. The basic elements in a graph are subgraphs which are defined in Definition 1.3.

Definition 1.3. A *subgraph* of a graph $\langle V, E, \phi, \psi \rangle$ is a graph $\langle V', E', \phi, \psi \rangle$, where $V' \subseteq V$ and $E' \subseteq E$.

According to Definition 1.3, a subgraph is also a graph. Therefore, in this thesis by default, all subgraphs are connected as well. A graph is a more generalized structure than a tree:

Definition 1.4. A *labeled* and *ordered tree* is a special case of a graph, where each node has only one incoming edge except for one *root* node which has no incoming edges.

Figure 1.1b shows a tree example. Each tree has a root node. We usually call that the tree is rooted at this node. Assuming (m, n) is an edge in a tree (namely, $m \rightarrow n$), m is called a *parent* or *head* while n is called a *child* or *dependent*. Nodes which have the same head are called *siblings*. Nodes which are reachable from a node n are *descendants* of n. In tree-based SMT, the basic translation units are usually subtrees which are defined as follows:

Definition 1.5. A *subtree* of a tree T is a tree T', which is rooted at a node n of T and includes all descendants of n.

A subtree is a special case of a treelet (Figure 1.1c) as defined in Definition 1.6.

Definition 1.6. A *treelet* is a connected subgraph of a tree.

1.2 Brief Overview of SMT

SMT starts from sequence-based models where the basic translation units are words or phrases. IBM made the first breakthrough on SMT by statistically modeling the translation process at the word-level (Brown et al., 1988; Brown et al., 1990; Brown et al., 1993). The well-known phrase-based translation model (Koehn et al., 2003) significantly improved upon word-based models by extending translation units from single words to phrases which allow local phenomena, such as word order, word deletion, and word insertion, to be captured. However, conventional phrase-based models are known to be weak at phrase reordering and learning generalizations (discontinuous phrases) such as Chinese *Yu* ... *WuGuan* to English *has nothing to do with*.

Source:	$X_{[1]}$ Yu $X_{[2]}$ WuGuan
Target:	$X_{[1]}$ has nothing to do with $X_{[2]}$

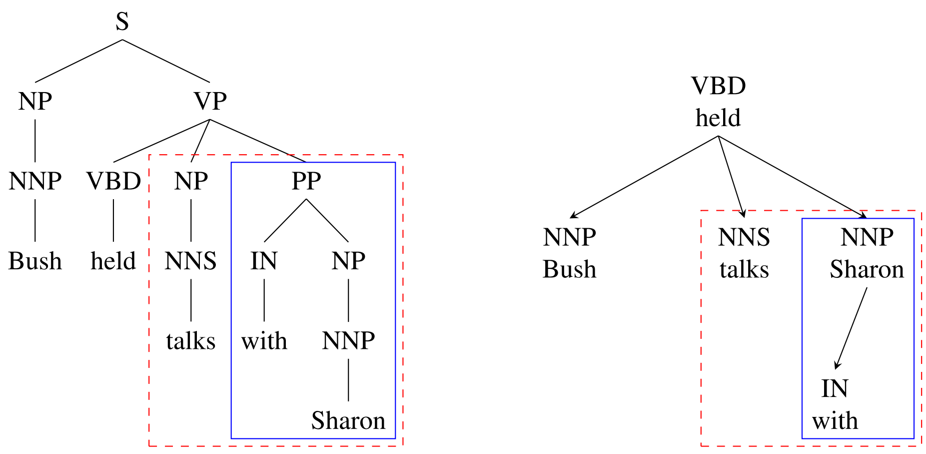
Figure 1.2: An example of a translation rule. X is a general non-terminal representing a gap. Indexes on X indicate mappings between source gaps and target gaps.

Therefore, tree-based (also called syntax-based) models were proposed to learn recursive translation rules. Chiang (2005) and Chiang (2007) proposed a formally syntax-based model. In this model, translation rules can be automatically learned from sentence pairs without linguistic annotations. These translation rules allow gaps which are represented by non-terminals and can be filled with other rules. Mappings between source and target non-terminals indicate how target phrases are reordered. Figure 1.2 shows an example of a translation rule. However, because the model only uses one general non-terminal X, it is hard to decide which rule should be used to replace a gap during translating a sentence.

Compared with *formally* syntax-based models, *linguistically* syntax-based models[1] make use of linguistic annotations and are believed to be better choices. One of the most widely used linguistic structures is constituent trees (as shown in Figure 1.3a) which provide syntactic categories and hierarchies of components (or phrases) in a sentence. However, translation models based on constituent structures (Galley et al., 2006; Liu et al., 2006; Huang et al., 2006a; Nesson et al., 2006; Zhang et al., 2007) could be constrained by rules which allow internal structures and linguistic annotations, because these rules may be too specific to be used during decoding (Koehn, 2010). In addition, these models usually focus on linguistically well-formed phrases (syntactic phrases, exactly covered by subtrees), such as the phrase *with Sharon* in Figure 1.3a which is covered by the subtree rooted at the node *PP*. Therefore, non-syntactic phrases, such as *talks with Sharon* in Figure 1.3a, are ignored.

Different from constituent trees, dependency trees (as shown in Figure 1.3b) directly

[1] Following Chiang (2005), in this thesis, tree-based models are distinguished according to whether they make use of grammars and linguistic annotations: formally syntax-based models are based on synchronous grammars, while linguistically syntax-based models are defined over structures informed by syntactic theory. A model can be both formally syntax-based and linguistically syntax-based when it uses both synchronous grammars and linguistic annotations.



(a) Constituent Tree

(b) Dependency Tree

Figure 1.3: Comparison of a constituent tree and a dependency tree of an English sentence. Following convention, edge directions are ignored in the constituent tree. The phrase *with Sharon* in solid rectangles is an example of a syntactic phrase, while the phrase *talks with Sharon* in dashed rectangles is an example of a non-syntactic phrase.

model syntactic and/or semantic relations between words in a sentence. The property enables dependency-based models to use and cover both syntactic phrases (Xie et al., 2011), such as *with Sharon* in Figure 1.3b, and non-syntactic phrases connected in trees (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007), such as *Bush held talks*. However, non-syntactic phrases which are not connected in dependency trees, such as *talks with Sharon* in Figure 1.3b, are not considered.

1.3 Why Graphs And Which Type?

Although syntactic phrases are more reliable in quality and have linguistic meaning, discarding other phrases is a harsh decision, which usually does not work well in practice, as non-syntactic phrases can be quite useful to improve rule coverage and are extremely important to system performance (Koehn et al., 2003). Therefore, researchers have tried to patch tree-based models by using extended labels to cover non-syntactic phrases (Marcu et al., 2006; Almaghout et al., 2011; Almaghout et al., 2012). However, such methods do not change the fundamental linguistic theories and grammars used in their models. This suggests that they may still have a weaker generative capacity over structures. In this sense, how to naturally handle non-syntactic phrases is still one of the major challenges in SMT.

The difficulty of handling non-syntactic phrases in tree-based models is mainly due to the fact that trees are recursive structures where subtrees rooted at sibling nodes do not overlap with each other, and thus phrases in trees are naturally divided into different groups: syntactic or non-syntactic. Even though dependency-based models could alleviate this by taking connected subgraphs into consideration (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007; Chen et al., 2014), the number of subgraphs is still limited in dependency trees and thus plenty of phrases are not considered.

An obvious observation is that non-syntactic phrases which are not connected in trees could be connected in terms of sequential structures, such as the phrase *talks with* in Figure 1.3b. In this sense, the two structures complement one another. Since both trees and sequences are special cases of graphs, a possible way of integrating non-syntactic phrases into tree-based models is using graphs where the basic units are subgraphs and thus phrases have no syntactic types and are not distinguished.

Therefore, in this thesis, we explore the possibility of constructing graphs and design graph-based translation models which translate graphs into strings. We take subgraphs which are connected in graphs as the basic translation units. Compared with phrases or subtrees, these subgraphs are more flexible which may cover either syntactic or non-syntactic phrases in terms of tree structures.

Since dependency trees have the best inter-lingual phrasal cohesion property, i.e. phrases in one language tend to stay together during translation (Fox, 2002), and provide the flexibility of covering different types of phrases, in this thesis we construct graphs based on them. We call these graphs **dependency graphs**. In order to build graphs where non-syntactic phrases are connected, we add sequential relations into dependency trees. Figure 1.4 shows an example of a graph where an additional edge (in dash) is added to the dependency tree in Figure 1.3b so that the non-syntactic phrase *talks with Sharon* is connected in the graph. The edge is added because *talks* and *with* are adjacent words in the sequence. The

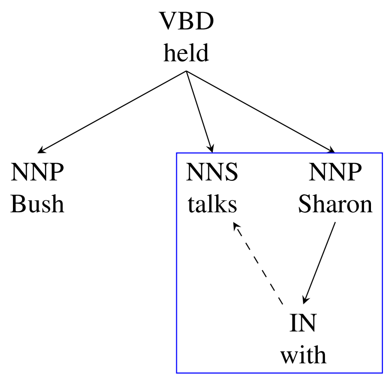


Figure 1.4: An example of a graph where an additional edge (dashed arrow) is added to a dependency tree so that the non-syntactic phrase *talks with Sharon* (in the rectangle) is connected in the graph.

advantages of using dependency graphs are as follows:

- Compared with using hypergraphs in translation models (Jones et al., 2012), we can easily build a large training corpus by parsing sentences into trees and then adding sequential relations.
- These graphs encode both local relations from sequences and long-distance relations from trees and provide more flexible translation units than sequences and trees.
- Models based on these graphs can use both syntactic and non-syntactic phrases without distinction as long as they are connected in the graphs.

1.4 Research Questions

Because both dependency relations and sequential relations are included in dependency graphs, they provide more flexible translation units than dependency trees alone. Therefore, we would like to know whether such kinds of units improve a dependency tree-to-string model. This leads to our first research question:

> **RQ1** *Can we improve dependency tree-to-string translation models by incorporating more translation units implied by sequential relations?*

In **RQ1**, we will examine two kinds of sequential relations: (i) **bigram relations** between two continuous words which make all non-syntactic phrases available. For example, both *held talks with* and *talks with Sharon* in Figure 1.3b are implied by bigram relations because they are continuous phrases; and (ii) **sibling relations** between siblings. For example, in Figure 1.3b the three words *Bush*, *talks*, and *Sharon* are siblings because they have the same head word *held*. Even though the concept of siblings only exists in a tree structure, we treat it as a sequential relation because it will be used to connect subtrees and make non-syntactic phrases available. For example, the relation between *talks* and *Sharon* in Figure 1.3b implies the availability of the phrase *talks with Sharon*. Although, compared with bigram relations, sibling relations allow fewer phrases to be considered, the phrases connected by them are more linguistically motivated.

Following **RQ1**, we need to consider how to construct graphs which uniformly encode both dependency relations and sequential relations, and where phrases implied by these relations are connected. This leads to our second research question:

> **RQ2** *Can we construct dependency graphs which combine dependency relations and sequential relations in a unified representation?*

Assuming the availability of graphs, how to translate them is another challenge. Inspired by phrase-based MT and treelet-based MT (Menezes and Quirk, 2005; Quirk et al., 2005) which segment input sequences or trees, we explore the possibility of taking connected subgraphs as the basic translation units in our graph-based translation model. One of the advantages of using subgraphs is that a subgraph may cover either a discontinuous phrase or a continuous phrase without distinction. Therefore, our next research question is as follows:

> **RQ3** *Can we translate input graphs into target sentences by graph segmentation where subgraphs, which may cover discontinuous phrases, are the basic translation units?*

However, graph-based models using graph segmentation will have a similar reordering problem as in the phrase-based models. Inspired by using grammars for tree-based MT

Structure	Segmentational	Recursive
Sequence	• word-based models (Brown et al., 1988; Brown et al., 1990; Brown et al., 1993) • phrase-based models (Koehn et al., 2003)	–
Tree	• dependency treelet-to-string models (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007) • dependency edge-transfer models (Chen et al., 2014)	• hierarchical phrase-based models (Chiang, 2005; Chiang, 2007) • tree-to-string models (Liu et al., 2006; Huang and Mi, 2010) • string-to-tree models (Galley et al., 2006; Marcu et al., 2006) • tree-to-tree models (Nesson et al., 2006; Zhang et al., 2007) • string-to-dependency models (Shen et al., 2010) • dependency-to-string models (Xie et al., 2011, **RQ1**)
Graph	segmentational graph-to-string model (**RQ2**&**RQ3**)	• semantic-based models (Jones et al., 2012) • recursive graph-to-string models (**RQ2**&**RQ4**)

Table 1.1: Comparison between our work and existing models in terms of two dimensions: (i) structures, including sequence, tree, and graph; and (ii) translation techniques, including segmentational and recursive.

to learn recursive translation rules, we further investigate the possibility of using graph grammars in our graph-based models. Accordingly, our final research question is:

> **RQ4** *Can we translate graphs by using graph grammars which parse the graphs and simultaneously generate target sentences using recursive translation rules?*

Table 1.1 compares our work with sequence-based and tree-based models which will be introduced in Chapter 2. We divide various models into groups according to structures (sequences, trees, and graphs) and translation techniques (segmentational and recursive) they use. *segmentational* means that input structures are segmented into units and a complete translation is obtained by combining translations of these units, e.g. the phrase-based model. By contrast, *recursive* denotes that inputs are translated by using recursive rules and a complete translation is obtained by recursively combining translations of lower-level structures, e.g. the hierarchical phrase-based model.

As shown in Table 1.1, in **RQ1** we will improve a dependency tree-to-string (also called dependency-to-string in this thesis) model. In **RQ3** we will present a graph-to-string model based on graph segmentation, while in **RQ4** we will translate graphs into strings based on graph grammars. **RQ2** provides methods of constructing dependency graphs which will be used subsequently in **RQ3** and **RQ4**. Table 1.2 compares our models with each other in detail which will be introduced in Chapters 3–5.

Note that our models use graph structures on the source side and translate graphs into strings. This separates our models from others which use structures on the target side, e.g. string-to-tree models (Galley et al., 2004; Galley et al., 2006; Marcu et al., 2006) and string-to-dependency models (Shen et al., 2010). During decoding, string-to-tree models translate a source sentence into a target tree which is constructed *during* decoding. By contrast, we construct source graphs *before* decoding. Our models translate these graphs into target sentences by matching rules with them.

	Dep2Str (**RQ1**)	SegGBMT (**RQ2**)	SNRG and SERG (**RQ4**)
Graph used	dependency tree	dependency-bigram graph	dependency-edge graph, dependency-sibling graph
Non-terminals	yes	no	yes
Rule source			
connected	yes	yes	yes
gaps	yes	yes	yes
continuity	yes	no	yes
Rule target			
gaps	yes	no	yes
continuity	yes	yes	yes

Table 1.2: Comparison between our translation models in three research questions, in terms of: graph structures used, whether rules use non-terminals, whether rule source is connected, whether rule source and target allow gaps, and whether rule source and target are required to cover continuous spans during decoding.

1.5 Thesis Structure

The rest of the thesis is structured as follows:

- In Chapter 2, we review the background and history of SMT. We will first introduce three kinds of statistical models: sequence-based models, tree-based models, and graph-based models. We will describe how these models are defined and their decoding algorithms for generating translations. Then, we introduce our experimental set-up, including data, tools, tuning algorithms, and automatic evaluation metrics. We will also build two baseline systems for comparison.

- In Chapter 3, we firstly review a dependency tree-to-string model and then improve it by using structures or phrases indicated by sequential relations which are useful to construct graphs. To achieve this, we propose a pseudo forest-to-string model which decomposes a dependency structure into smaller pieces. Such decomposition allows phrases implied by sibling relations to be covered. In experiments, we further integrate phrasal rules which are implied by bigram relations into our model by mixing them with other translation rules.

- In Chapter 4, we construct graphs which combine dependency relations and bigram relations. We present a segmentational graph-based model which segments a graph into subgraphs. Translations can then be obtained by combining translations of each subgraph left-to-right. Furthermore, in order to take source context into consideration during the graph segmentation process, we propose a graph segmentation model to help select a better subgraph to translate.

- In Chapter 5, we build graphs which take dependency relations and sibling relations into consideration. Then, we use synchronous graph grammars to translate these graphs. The advantage of using synchronous grammars is that the learned recursive rules directly encode reordering information. In this chapter, two kinds of graphs are built: one is edge-labeled where sibling edges are connected, while the other is node-labeled where sibling links are directly added to dependency trees.
- We conclude in Chapter 6 with a summary of our work and contributions of the thesis. Then, we present avenues for future work.

1.6 Related Publications

The published papers related to this thesis are as follows:

1. Liangyou Li, Andy Way, and Qun Liu (2016). Graph-Based Translation Via Graph Segmentation. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 97–107.
2. Liangyou Li, Andy Way, and Qun Liu (2015). Dependency Graph-to-String Translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 33–43.
3. Liangyou Li, Jun Xie, Andy Way, and Qun Liu (2014). Transformation and Decomposition for Efficiently Implementing and Improving Dependency-to-String Model In Moses. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 122–131.

It is remarkable that a science which began with the consideration of games of chance should have become the most important object of human knowledge.

— Pierre-Simon Laplace

Chapter 2

Statistical Machine Translation

Typically, the first step of building an SMT system is (i) learning models from parallel corpora, which are defined over $p(t|s)$ to score each translation. A parallel corpus consists of texts in one language and their translations in another language. Table 2.1 shows examples of sentence pairs from a parallel corpus.

After obtaining models, we need to: (ii) tune parameters for these models so that good translations obtain relatively higher probabilities while bad translations obtain relatively lower probabilities; (iii) decode a source sentence to find its translation t^* in the target language so that $t^* = \arg\max_t p(t|s)$; (iv) evaluate translation performance of the system.

In this thesis, we focus on the definition of $p(t|s)$, namely statistical models. Depending on what kind of structural information is used, we divide different models into three

Chinese	English
BuShi Yu ShaLong JuXing Le HuiTan	Bush held talks with Sharon
BoLiWeiYa JuXing ZongTong Yu GuoHui XuanJu	Bolivia holds presidential and parliament elections
2020Nian ShiJieBei Zai NanFei ChengGong JuXing	2010 world cup was held successfully in South Africa

Table 2.1: Examples of Chinese–English sentence pairs in a parallel corpus.

categories: sequence-based models, tree-based models, and graph-based models. We will firstly explain these models which have been explored in previous work. Then, we describe experimental settings in this thesis, including data, tools, tuning algorithms, evaluation metrics, and baseline systems.

2.1 Sequence-Based Models

Sequence-based models use sequential information of sentences to build models. The atomic translation units in sequence-based models are word sequences (or phrases). A final translation can be obtained by combining translations of each source phrase. These models include word-based models (Brown et al., 1988; Brown et al., 1990; Brown et al., 1993) and phrase-based models (Och et al., 1999; Koehn et al., 2003).

2.1.1 Word-Based Models

Word-based models were firstly introduced by IBM in the late 1980s. Thereafter, during the 2000s research in the field of SMT grew rapidly. Word-based models mathematically formulate SMT as a series of word-to-word translation. Given a parallel corpus, word-based models can learn word-translation probabilities, as illustrated in Table 2.2.

Word Alignment

A fundamental concept in word-based models is **word alignment**, which is a function defining many-to-many mappings from source words to target words in a sentence pair.

Source Word	Target Word	Probability
BuShi	Bush	0.7
	president	0.2
	US	0.1
Yu	and	0.6
	with	0.4

Table 2.2: Examples of word translations and their probabilities in word-based models.

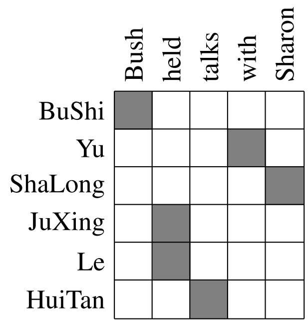


Figure 2.1: Illustration of word alignments from a Chinese sentence (left) to an English sentence (top) as indicated by gray marks.

However, in word-based models, the learned word alignment only includes many-to-one mappings. Word alignment is treated as a latent variable which can be learned iteratively by the Expectation-Maximization algorithm (Dempster et al., 1977). This is based on the idea that the alignment can be deduced from translation probabilities of words while word-translation probabilities can be learned from word alignments. Figure 2.1 illustrates a word-alignment matrix between a Chinese sentence and an English Sentence.

Noisy-Channel Model

Inspired by the success of using information theory in automatic speech recognition, Brown et al. (1988) apply Bayes' rule to $p(t|s)$ as in Equation (2.1):

$$p(t|s) = \frac{p(s|t)p(t)}{p(s)} \tag{2.1}$$

Since the probability $p(s)$ is identical for a given source sentence s, it has no impact on the selection of translations and thus can be ignored. Therefore, the task of finding the best translation can be formulated as in Equation (2.2):

$$\begin{aligned} t^* &= \arg\max_t p(t|s) \\ &= \arg\max_t p(s|t)p(t) \end{aligned} \tag{2.2}$$

Equation (2.2) divides $p(t|s)$ into two probabilities: $p(s|t)$ and $p(t)$. $p(s|t)$ is called the

translation model probability while $p(t)$ is called the **language model probability**. The way of combining the two probabilities is called the **noisy-channel model**.

IBM Models

Brown et al. (1993) proposed five models over $p(s|t)$. Model 1 assumes that the probability of a source word being produced by a target word only depends on the target word as in Equation (2.3):

$$p(s|t) = \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} p(s_j|t_i), \tag{2.3}$$

where m is the length of the source sentence s, l is the length of the target sentence t, and ε is a constant. However, in Model 1 source words can be arbitrarily aligned to any target words. So Model 2 explicitly models the alignment as in Equation (2.4):

$$p(s|t) = \varepsilon \prod_{j=1}^{m} \sum_{i=0}^{l} p(s_j|t_{a_j}) a(a_j|j,m,l), \tag{2.4}$$

where $a(a_j|j,m,l)$ denotes the probability of a source word at the position j being aligned with a target word at the position a_j given the source sentence length m and target sentence length l. Because Model 2 does not explicitly model many-to-one alignments, Model 3 introduces a fertility probability $n(\phi_i|t_i)$ which models the case that one target word t_i corresponds to the number of ϕ_i source words. Model 3 also changes the alignment probability a into a distortion model $d(j|a_j,m,l)$ which can be seen as a simple reordering model. Model 4 refines Model 3 by replacing the position-based distortion model with a class-based distortion model. Model 5 refines Model 4 by addressing a deficiency where probabilities are assigned to impossible events.

Language Model

The language model $p(t)$ is an essential component in SMT systems as it can help to find a more fluent translation by considering word order on the target side. Given a sentence

$t = (w_1, w_2, \cdots, w_l)$, the language model can be formulated as in Equation (2.5):

$$p(w_1, w_2, \cdots, w_l) = \prod_{i=1}^{l} p(w_i | w_1, \cdots, w_{i-1}) \tag{2.5}$$

Standard SMT systems use n-gram language models which are based on the assumption that the probability of a word only depends on the previous $n-1$ words, as in Equation (2.6).

$$p(w_1, w_2, \cdots, w_l) = \prod_{i=1}^{l} p(w_i | w_{i-n+1}, \cdots, w_{i-1}) \tag{2.6}$$

where

$$p(w_i | w_{i-n}, \cdots, w_{i-1}) = \frac{\text{count}(w_{i-n+1}, \cdots, w_{i-1}, w_i)}{\sum_w \text{count}(w_{i-n+1}, \cdots, w_{i-1}, w)} \tag{2.7}$$

2.1.2 Phrase-Based Models

Phrase-based models extend word-based models by translating phrases instead of single words. In this case, a phrase means only a sequence of continuous words in a sentence, which is not necessarily related to the linguistic notion of a phrase. Different from word-based models where many-to-one word mappings exist, phrase-based models imply a one-to-one phrase alignment, as illustrated in Figure 2.2. Thanks to the capability of capturing local phenomena, translating word groups instead of single words can help to solve translation ambiguities. Therefore, since the 2000s phrase-based models have replaced word-based models as the state-of-the-art in SMT.

The word alignment produced by word-based models only considers one direction, namely a target word is aligned to multiple source words. Instead, phrase-based models

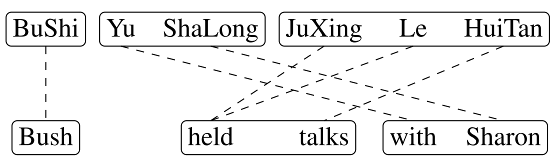


Figure 2.2: An illustration of a one-to-one phrase alignment in a phrase-based model. Dashed lines indicate word alignments.

use a refined word alignment which is obtained by symmetrizing IBM models in both directions (Och and Ney, 2003). The refined word-alignment model consists of many-to-many mappings between words. This means multiple source words can be aligned to one target word and vice versa.

Model Definition

Given a pair of sentences $\langle s = \overline{s}_1 \cdots \overline{s}_I, t = \overline{t}_1 \cdots \overline{t}_I \rangle$, in the conventional phrase-based model, $p(s|t)$ is defined as in Equation (2.8):

$$p(\overline{s}_1^I | \overline{t}_1^I) = \prod_{i=1}^{I} p(\overline{s}_{a_i} | \overline{t}_i) d(\overline{s}_{a_i}, \overline{s}_{a_{i-1}}) \tag{2.8}$$

The target sentence t is broken into I phrases $\overline{t}_1 \cdots \overline{t}_I$, each of which is a translation of a source phrase $\overline{s}_{a_i}$. The distortion function d is a simple reordering model which is based on the distance between the start position of $\overline{s}_{a_i}$ and the end position of $\overline{s}_{a_{i-1}}$.

The performance of phrase-based models relies on the quality of phrase pairs in a translation table, which consists of all phrase pairs extracted from a parallel corpus associated with translation probabilities. Table 2.3 shows an example of a translation table with probabilities $p(\overline{t}|\overline{s})$. Conventionally, a phrase pair $\langle \overline{s}, \overline{t} \rangle$ has two properties: (i) $\overline{s}$ and $\overline{t}$ are continuous phrases; and (ii) $\langle \overline{s}, \overline{t} \rangle$ is consistent with a word alignment a (Och and Ney, 2004):

$$\forall (i,j) \in a, s_i \in \overline{s} \Leftrightarrow t_j \in \overline{t} \text{ and } \exists s_i \in \overline{s}, t_j \in \overline{t}, (i,j) \in a.$$

which means no word is aligned to another word outside the phrase pair. For example, given

Source Phrase	Target Phrase	Probability
BuShi	Bush	0.5
	president Bush	0.3
	the US president	0.2
BuShi Yu	Bush and	0.7
	the president and	0.3

Table 2.3: Examples of phrase translations and their probabilities in a phrase-based model.

the word-aligned sentence pair in Figure 2.2, we can extract a phrase pair $\langle$*JuXing Le*, *held*$\rangle$, but we cannot extract $\langle$*JuXing*, *held*$\rangle$ because the target word *held* is aligned to an outside source word *Le*.

Because phrase boundaries are unavailable, phrase-based models involve jointly segmenting a source sentence into phrases and translating the phrases. The decoding objective can be formulated as a sum over all possible derivations:

$$t^* = \arg\max_t \sum_d p(s,d|t)p(t) \tag{2.9}$$

where $d \in D(s,t)$ is a derivation which implies a segmentation of s and generates a target sentence t. However, the summation in Equation (2.9) is computationally intractable (Williams, 2014). Thus, conventionally the objective is approximated as finding the best derivation, as in Equation (2.10):

$$t^* = \arg\max_{t,d} p(s,d|t)p(t) \tag{2.10}$$

Log-Linear Model

Och and Ney (2002) proposed a more general framework to replace the noisy-channel model. Following the maximum entropy model (Berger et al., 1996), the probability of a translation can be formulated as in Equation (2.11):

$$p(t|s) = \frac{\exp\{\sum_{i=1}^{M} \lambda_i h_i(s,t)\}}{\sum_{t'} \exp\{\sum_{i=1}^{M} \lambda_i h_i(s,t')\}}, \tag{2.11}$$

where $h_i(s,t)$ are feature functions defined over the source s and the target t, λ_i are feature weights, and M is the number of feature functions considered. By ignoring the denominator as it is constant for a given source sentence, the translation can be obtained by Equation (2.12):

$$t^* = \arg\max_t \sum_{i=1}^{M} \lambda_i h_i(s,t) \tag{2.12}$$

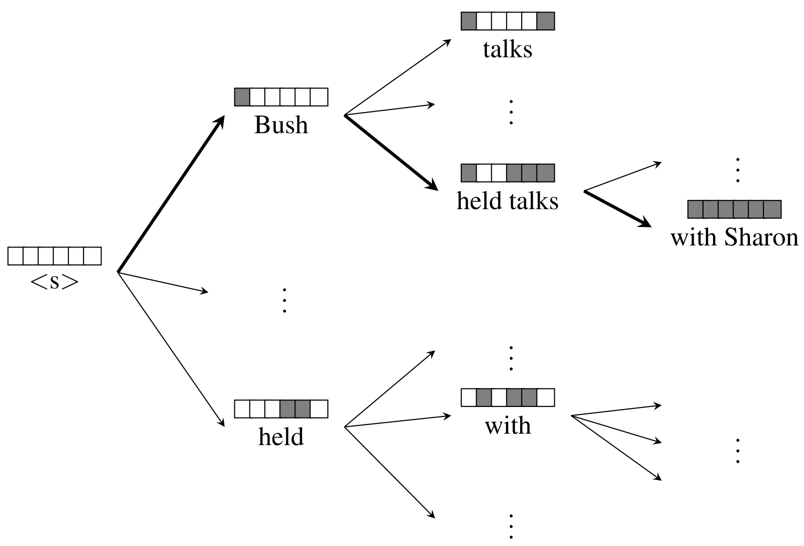


Figure 2.3: Illustration of the search space in phrase-based models to translate a Chinese sentence *BuShi Yu ShaLong JuXing Le HuiTan*. When a node is expanded by a phrase translation, the corresponding source positions in the coverage vector are marked as translated (in gray). The path in bold implies a complete translation.

As Och and Ney (2002) pointed out, the noisy-channel model is a special case of Equation (2.12) (when $M = 2$, $\lambda_1 = \lambda_2 = 1$, $h_1 = \log p(s|t)$ and $h_2 = \log p(t)$). One of the advantages of the log-linear model is that it can integrate an arbitrary number of feature functions whose weights are tunable. These features usually can significantly improve a system.

Decoding Based on Beam Search

Phrase-based decoders generate hypotheses (partial translations) from left to right. Each hypothesis maintains a **coverage vector** to indicate which source words have been translated so far. A hypothesis can be extended on the right by translating an uncovered source phrase. The translation process ends when all source words have been translated.

Figure 2.3 shows a search space, which is represented in a tree structure, of translating a sentence from Chinese to English using a phrase-based model. Translation begins with the start symbol <s>. When a node is extended by a translation of an uncovered source phrase, the corresponding positions in the coverage vector are then marked as translated. A path

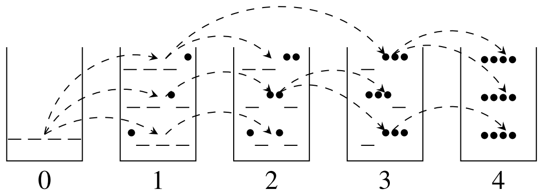


Figure 2.4: Beam search for phrase-based models (Liu and Huang, 2014). Hypotheses covering the same number of source words (integers under stacks) are grouped in the same stack. • denotes a covered source position while _ indicates an uncovered position.

from the start node to a leaf node implies a translation.

However, the time complexity of an exhaustive search in the search space is exponential to the input length (Knight, 1999). Therefore, beam search (as in Figure 2.4) is usually taken as an approximate search strategy to reduce the size of the decoding space. Hypotheses which cover the same number of source words are grouped in a stack where hypotheses can be pruned according to their partial translation cost and an estimated future cost.

2.1.3 Shortcomings of Sequence-Based Models

One of the significant drawbacks of sequence-based models lies in reordering words. Although phrase-based models extend word-based models by considering word groups which capture word order within phrases, they cannot reorder the phrases themselves. Therefore, by default, a distance-based reordering function as in Equation (2.8) is adopted. More sophisticated reordering models (Koehn et al., 2005; Xiong et al., 2006; Galley and Manning, 2008; Cherry, 2013) can also be used to build a stronger phrase-based system. However, sequence-based models are still known to be weak at long-distance reordering.

Another disadvantage of conventional sequence-based models is that only continuous phrases are considered, and thus the learned translation pairs cannot be generalized even though sometimes an apparent pattern can be recognized. Galley and Manning (2010) extend conventional phrase-based models by allowing phrases with gaps (discontinuous phrases). However, without using linguistic knowledge, the model can learn plenty of unreliable translation rules resulting in a huge model and a slower system.

2.2 Tree-Based Models

Tree-based (or syntax-based) models are proposed to solve problems in sequence-based models by learning translation rules which allow phrase reordering and generalization. Based on which kind of syntax structures is used, we explain tree-based models in three categories: Hierarchical Phrase-Based (HPB) models (Chiang, 2005; Chiang, 2007), constituent tree-based models (Galley et al., 2004; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006a; Nesson et al., 2006; Zhang et al., 2007), and dependency tree-based models (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007; Xie et al., 2011).

2.2.1 Hierarchical Phrase-Based Models

A hierarchical phrase is an extension of a phrase by allowing gaps where other hierarchical phrases are embedded. Recursively using hierarchical phrases produces a tree structure where no linguistic annotations are included. The HPB model is a formally syntax-based model, which is formulated by a Synchronous Context Free Grammar (SCFG) (Lewis and Stearns, 1968), as in Definition 2.1.

Definition 2.1. An SCFG is a tuple $\langle N, T, T', P, S\rangle$, where

- N is a finite set of non-terminal symbols.
- T and T' are finite sets of terminal symbols.
- $S \in N$ is the start symbol.
- P is a finite set of productions of the form $(A \rightarrow \langle R, R', \sim\rangle)$, where $A \in N$, R is a *sequence* over $N \bigcup T$ and R' is a *sequence* over $N \bigcup T'$. $\sim$ is a one-to-one mapping between non-terminal symbols in R and R'.

Model Definition

In HPB model, gaps are represented by a generic non-terminal symbol X. Therefore, rules in the HPB model are in the form of (2.13):

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle, \tag{2.13}$$

where γ is a string over source terminal symbols and non-terminals, α is a string over target terminal symbols and non-terminals, and $\sim$ is a one-to-one mapping between non-terminals in γ and α and indicates reordering information. An example of a recursive rule is as in (2.14):

$$X \rightarrow \langle \text{ BuShi } X_{[1]} \text{ JuXing Le } X_{[2]}, \text{Bush held } X_{[2]}\ X_{[1]} \rangle, \tag{2.14}$$

where indexes on non-terminals indicate the mappings. Note that, when no non-terminals exist in α and γ, rules are used to translate phrases as in the phrase-based model, such as (2.15).

$$X \rightarrow \langle \text{Yu ShaLong, with Sharon} \rangle, \tag{2.15}$$

All HPB rules can be automatically learned from word-aligned sentence pairs. A rule extractor firstly extracts rules without non-terminals which will be subsequently used to produce recursive rules by replacing phrase pairs inside them with non-terminals. Figure 2.5 illustrates how to extract the rule (2.14) from a sentence pair according to a given word alignment, where the phrase pair ⟨*Yu ShaLong, with Sharon*⟩ is replaced by $X_{[1]}$ while the phrase pair ⟨*HuiTan, talks*⟩ is replaced by $X_{[2]}$.

In addition to translation rules above, two glue rules are used for robustness:

$$S \rightarrow \langle S_{[1]} X_{[2]}, S_{[1]} X_{[2]} \rangle \tag{2.16a}$$

$$S \rightarrow \langle X_{[1]}, X_{[1]} \rangle \tag{2.16b}$$

where S is the start symbol. Glue rules segment a sentence into a sequence of phrases which

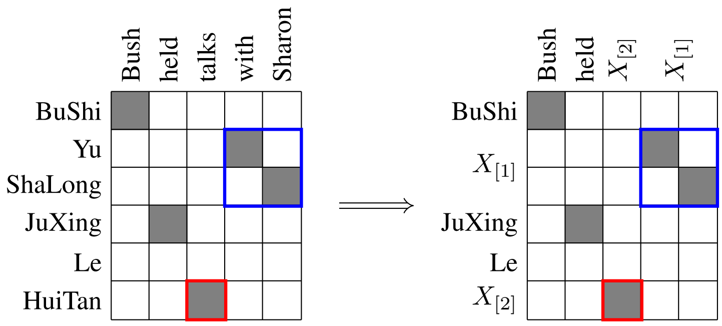


Figure 2.5: Illustrating the extraction of a hierarchical phrase-based rule. Two phrase pairs (in bold rectangles) are replaced by the general non-terminal X. Mappings between source (left) and target (top) non-terminals are indicated by indexes.

will be translated separately, and then their translations are combined without reordering. With the help of glue rules, we can make sure to obtain at least one translation of any sentence.

Similar to phrase-based models, the probability $p(t|s)$ is approximated by $p(d)$ as in Equation (2.17) which follows the log-linear model:

$$p(d) \propto \prod_{i=1}^{M} \phi_i(d)^{\lambda_i} \tag{2.17}$$

where $d = r_1 r_2 \cdots r_n$ is a derivation consisting of a sequence of rules r_j, ϕ_i are feature functions while λ_i are feature weights.[1] The best translation t^* can be obtained by searching for the best derivation, as in Equation (2.18):

$$t^* = \arg\max_{t,d} p(d) = \arg\max_{t,d} \sum_{i=1}^{M} \lambda_i \log \phi_i(d) \tag{2.18}$$

Chart Decoding

The best translation in HPB models can be obtained via chart parsing using the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970). Chart decoders generate hypotheses bottom-up. Each hypothesis is a translation of a phrase which covers

[1]Note that h_i in Equation (2.12) equals to $\log \phi_i$ in Equation (2.18). In the rest of the thesis, by default feature functions refer to ϕ_i.

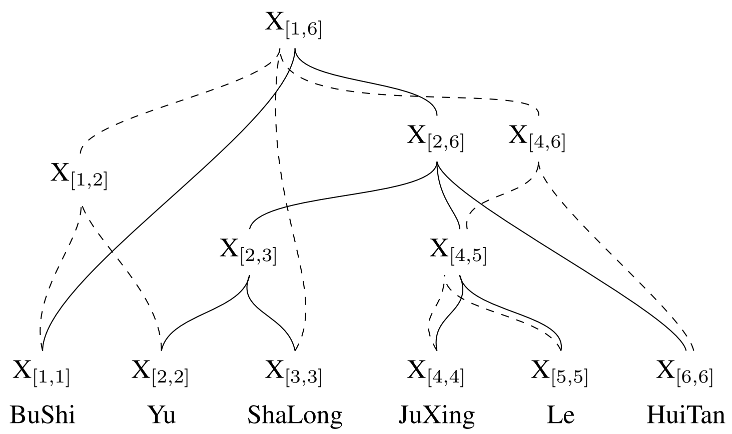


Figure 2.6: A derivation hypergraph consisting of two derivations marked by different types of lines. Edges indicate rules. Each node denotes a stack which covers a source span $[i, j]$ and has a non-terminal X. Different translations of the same source span are organized into the same stack.

a continuous span of an input sentence. Translations of smaller spans can be combined to produce translations of large spans by using translation rules which contain non-terminals. Non-terminals associated with their mappings in these rules work as the function of placeholders which specify the positions and order of smaller translations.

The search space of chart decoding can be represented by a hypergraph which contains all possible derivations, as illustrated in Figure 2.6. Each node represents a set of translations of a source span. Each edge in the hypergraph implies a rule application. Similar to phrase-based decoders (cf. Figure 2.4), hypotheses which cover the same source span are organized into the same stack so that they can be pruned and recombined.

Applying a rule which contains two non-terminals involves a combination of translations from two stacks. Assuming we have m such rules covering the same source span and each stack stores n translations, to obtain translations of the span, the decoder needs to consider all mn^2 possibilities. In practice, such an exhaustive search slows systems down, so cube pruning (Chiang, 2007) is used to quickly combine translations from multiple stacks, where rules and chart translations are sorted and then dynamically combined.

Integration of Linguistic Knowledge

Due to the lack of linguistic knowledge, the only non-terminal X often makes the HPB model hard to select the most appropriate rule. Therefore, some work refines this non-terminal using linguistic information, such as syntactic categories from constituent structures (Zollmann and Venugopal, 2006), Part-of-Speech (POS) tags or word classes (Zollmann and Vogel, 2011), Combinatory Categorial Grammar (CCG)-based supertags (Almaghout et al., 2011; Almaghout et al., 2012), and head information from dependency structures (Li et al., 2012a).

2.2.2 Constituent Tree-Based Models

A constituent (or phrasal) structure displays the functional components of a sentence. Based on which side is parsed into such a syntactic tree, models are categorized into three groups: string-to-tree models, tree-to-string models, and tree-to-tree models. Typically, these models can be formulated in a Synchronous Tree Substitution Grammar (STSG) (Eisner, 2003), as in Definition 2.2. We found that while R and R' in P in Definition 2.1 are strings, in Definition 2.2 they are trees. STSG can be seen as an extension of SCFG by considering multi-level tree structures and thus has a stronger generative capacity over tree pairs (Chiang, 2012).

Definition 2.2. An STSG is a tuple $\langle N, T, T', P, S\rangle$, where

- N is a finite set of non-terminal symbols.
- T and T' are finite sets of terminal symbols.
- $S \in N$ is the start symbol.
- P is a finite set of productions of the form $(A \to \langle R, R', \sim\rangle)$, where $A \in N$, R is a *tree* over $N \bigcup T$ and R' is a *tree* over $N \bigcup T'$. $\sim$ is a one-to-one mapping between non-terminal symbols in R and R'.

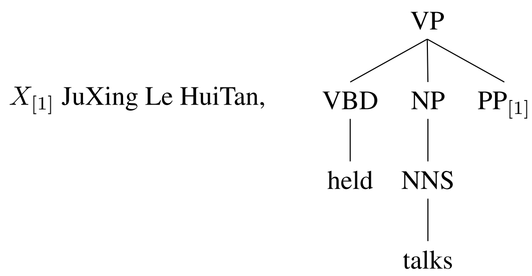


Figure 2.7: An example of a string-to-tree translation rule. X is the general non-terminal on the source side while *PP* and other non-leaf nodes are target non-terminals. Indexes on non-terminals indicate mappings between source and target non-terminals.

String-to-Tree Models

String-to-tree models can be traced back to Yamada and Knight (2001) and Yamada and Knight (2002) who assumed that a source sentence is obtained by processing each node of a target tree using several operations, including insertion, reordering, and translation. The translation task is to recover the target tree given the source sentence. Galley et al. (2004) and Galley et al. (2006) proposed a stronger string-to-tree model. Given a word alignment, this model automatically extracts transfer rules from a source string and a target tree. These rules map source phrases into target tree fragments.

Figure 2.7 shows an example of a typical rule in a string-to-tree model. The source side of a rule is a string over source terminals and non-terminals while the target side is a tree structure consisting of target non-terminals and terminal leaves. Similar to the HPB model, these rules can be recursively extracted by replacing smaller rules with non-terminals, as illustrated in Figure 2.8.

Given a source sentence, based on the CYK algorithm, the decoding process involves in parsing the input bottom-up using hierarchical phrases on the source side of rules and simultaneously generating a target tree. Target strings can be simply read off of the leaves of target trees.

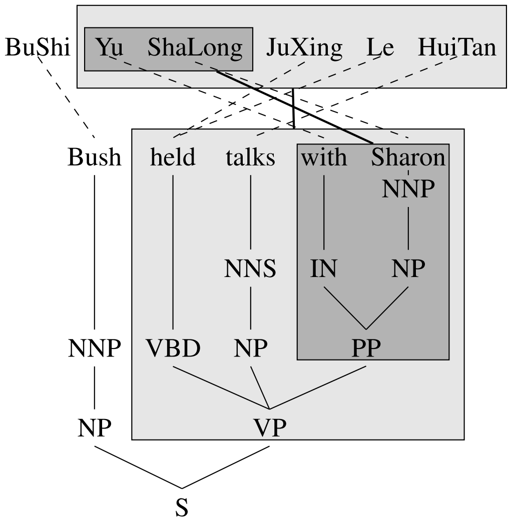


Figure 2.8: Illustrating the extraction of the string-to-tree rule in Figure 2.7 from a string–tree pair in lighter gray. The string–tree pair in darker gray is replaced by non-terminals. Dashed lines indicate the word alignment between source and target words.

Tree-to-String Models

Different from string-to-tree models, tree-to-string models are based on parse trees on the source side (Huang et al., 2006a; Huang et al., 2006b; Liu et al., 2006). At first, a source sentence is parsed into a constituent tree, and then translation rules, whose source sides are trees and target sides are strings, are extracted based on word alignments. During decoding, each node in the source tree is translated in a bottom-up manner. For a subtree rooted at a current node, the decoder uses a matched rule to translate the subtree into a target string, and non-terminals in the rule are replaced by translations of subtrees rooted at lower nodes. Compared with string-to-tree models, tree-to-string models can decode a sentence in linear time in practice with respect to the sentence length (Huang and Mi, 2010).

Tree-to-Tree Models

In a tree-to-tree model, both the source side and the target side of a rule are trees, as illustrated in Figure 2.9. Typically, two approaches are used in tree-to-tree models: (i) Given a source tree, transfer rules convert it to a target tree (Zhang et al., 2007); (ii) Given a source sentence, the decoder parses it and simultaneously produces a target tree (Nesson et al., 2006).

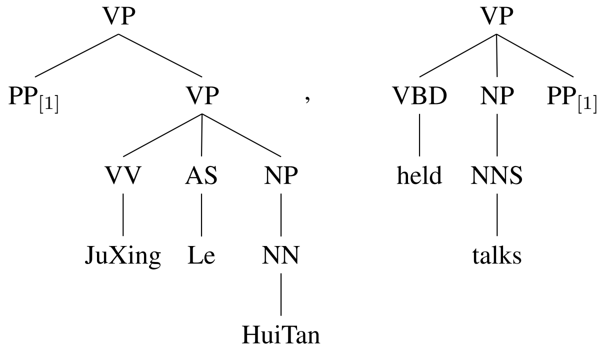


Figure 2.9: An example of a tree-to-tree translation rule. Indexes on non-terminals indicate mappings between source and target non-terminals.

One of the major challenges in a tree-to-tree model is the structural divergence between languages (Dorr, 1994), which is due to either systematic differences between two languages in expressing a concept syntactically or relatively free translations in the training corpora (Zhang et al., 2007).

2.2.3 Dependency Tree-Based Models

Dependency structures directly model relations between words in a sentence, each of which indicates the syntactic and/or semantic function of one word in relation to another word. Since dependency structures have the best inter-lingual cohesion property, it has been demonstrated to be helpful in SMT.

Shen et al. (2010) presented a model which is based on HPB models with the extension of a target dependency tree. The model focuses on well-formed dependency structures which are defined as fixed/floating structures. A fixed structure consists of a head and subtrees rooted at its children while a floating structure consists of subtrees rooted at consecutive sibling nodes. Figure 2.10 shows examples of fixed and floating structures in a dependency tree. Since both structures cover continuous spans, it is easier to integrate a dependency-based language model which scores a node according to its head and adjacent siblings and can significantly improve the system.

Another work on dependency-based translation is the treelet approach (Menezes and

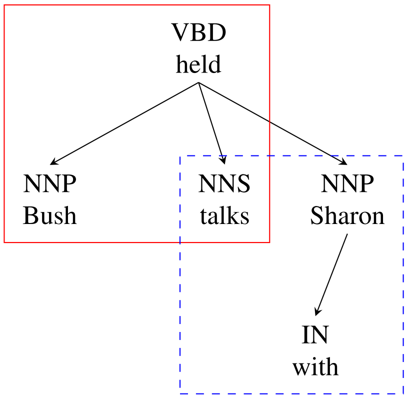


Figure 2.10: Examples of a fixed structure (in the solid rectangle) and a floating structure (in the dashed rectangle) in a dependency tree. While a fixed structure consists of a head node and subtrees rooted at the children of the head node, a floating structure only consists of subtrees rooted at sibling nodes. Both structures cover continuous spans of a sentence.

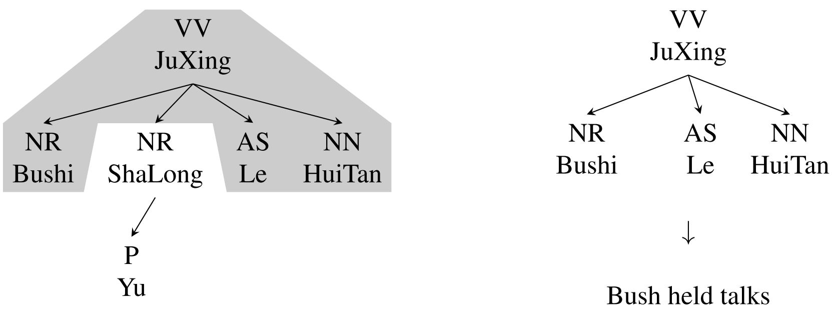
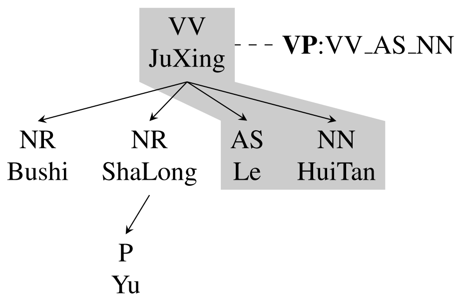


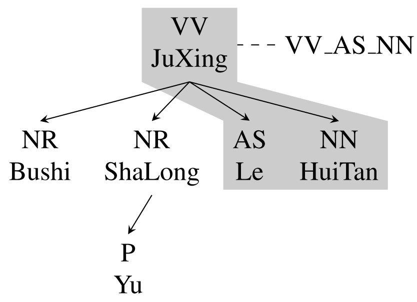
Figure 2.11: A dependency treelet-to-string translation rule (b) which can be applied to the treelet in gray in (a).

Quirk, 2005; Quirk et al., 2005), which utilizes dependency structures on the source side and projects them to the target side. Treelet-based decoders translate a dependency tree by bottom-up traversing the tree. Translations of a subtree can be obtained by combining translations of its disjoint treelets. Xiong et al. (2007) extended the treelet approach to allow gaps in treelets and used treelet-to-string rules. Figure 2.11 shows a rule to translate a dependency treelet into a target string. However, in these work, translation rules do not encode enough reordering information. Thus, another heuristic or separate reordering model is usually adopted to decide the positions of target words.

To encode reordering information, Xie et al. (2011) presented a dependency tree-to-



(a) Use non-syntactic phrases which are covered by constituent nodes (Meng et al., 2013). *VP:VV_AS_NN* is a special label to represent a non-syntactic phrase in shadow, where *VP* is a constituent label.



(b) Use non-syntactic phrases which are covered by fixed or floating structures (Xie et al., 2014). *VV_AS_NN* is a special label to represent a non-syntactic phrase in shadow which is covered by a fixed structure.

Figure 2.12: Two ways of creating labels to use non-syntactic phrases in a dependency tree

string model which is defined by a synchronous grammar similar to SCFG but including dependency relations. This model will be described in detail in Chapter 3. One of the significant weaknesses in the model is that it only considers syntactic phrases which are covered by complete subtrees. Since dependency trees are flatter than constituent trees, this model has a severe data-sparsity problem (Meng et al., 2013; Xie et al., 2014). To incorporate non-syntactic phrases, Meng et al. (2013) proposed to simultaneously use dependency trees and constituent trees so that phrases which are non-syntactic in dependency trees but syntactic in constituent trees can be covered. Figure 2.12a shows an example where a special label consisting of both a constituent label and POS tags of words is created to represent a non-syntactic phrase. By contrast, Xie et al. (2014) incorporated fixed and floating structures into the dependency tree-to-string model by creating special labels (as shown in Figure 2.12b) at run-time. However, to allow these special labels, both methods require a significant modification on the decoder. In addition, the number of non-syntactic phrases they use is very limited. In Chapter 3, we will present a simpler yet effective way to cover non-syntactic phrases.

2.2.4 Shortcomings of Tree-Based Models

When linguistic trees are used, translation performance of a system can be impacted by the accuracy of syntactic parsers (Quirk and Corston-Oliver, 2006). To alleviate this, several parse trees of a sentence can be compactly represented as a forest (Mi et al., 2008; Tu et al., 2010) which can then be used for translation. Joint parsing and translation (Liu and Liu, 2010) can also reduce the propagation of parsing errors.

Another significant challenge in linguistic tree-based models is integrating non-syntactic phrases which are not linguistically well-formed but can be important to translation performance of systems (Koehn et al., 2003; Hanneman and Lavie, 2009; Huck et al., 2014). To make use of such phrases, additional non-terminal symbols (Marcu et al., 2006; Zollmann and Venugopal, 2006; Almaghout et al., 2011; Almaghout et al., 2012) and binarization of syntax trees (Zhang et al., 2006; Wang et al., 2007) may be needed. However, such kinds of relaxation of syntactic constraints can result in less grammatical translations (Kaljahi et al., 2012).

Compared with constituent trees, dependency trees provide more flexible translation units, such as treelets or paths (Lin, 2004). However, sub-structures which are not connected in trees but which can significantly improve translation performance (Xie et al., 2014), such as the floating structure in Figure 2.10, are usually ignored.

By contrast, because the HPB model is not based on linguistically syntactic structures, it avoids the definition of linguistically syntactic and non-syntactic phrases and thus can make use of both without distinction. However, without linguistic annotations, the model usually is huge in size and has a problem with rule selection (He et al., 2008).

2.3 Graph-Based Models

Despite the use of syntax in tree-based models, it is still frequently unable to preserve basic meaning structures across languages. Therefore, as a more general and powerful representation than trees, graphs are introduced to better represent sentences. For instance,

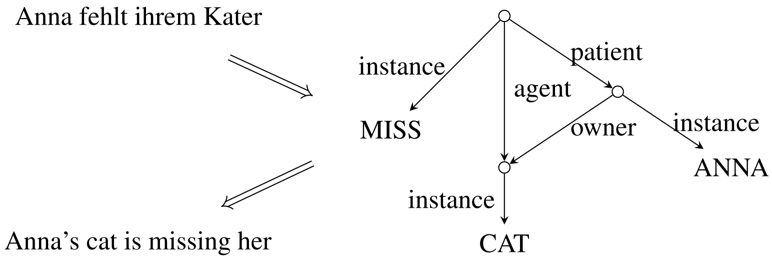


Figure 2.13: Illustration of semantic-based translation (Jones et al., 2012). A source sentence is firstly parsed into a hypergraph which is then converted to a target string.

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) uses hypergraphs to represent semantic meanings of sentences.

Jones et al. (2012) presented a semantics-based translation model, where the semantic meaning of a sentence is represented in a hypergraph. This model is based on a Synchronous Hyperedge Replacement Grammar (SHRG) (Drewes et al., 1997) and translates a sentence in two steps, as shown in Figure 2.13: (i) parsing a source string into a hypergraph using a source SHRG; and (ii) the 1-best source hypergraph is transformed into a target string using a target SHRG. However, the recognition algorithm for SHRG is in polynomial time but potentially of a high degree (Lautemann, 1990; Chiang et al., 2013). Furthermore, large parallel corpora annotated with hypergraphs are not readily available.

2.4 Experimental Set-up

In this thesis, we conduct large-scale experiments on two language pairs: Chinese–English (ZH–EN) and German–English (DE–EN). The two language pairs are selected because they have syntactically different word order, and thus long-distance reordering is much more important to translation performance of SMT systems. We hypothesize that dependency trees are suitable for the two language pairs as they directly encode syntactic and/or semantic relations between words, and dependency-based systems have the potential to perform long-distance reordering. Furthermore, the two language pairs have different linguistic characteristics which may have a different impact on our SMT systems:

- Chinese sentences have a larger Mean Dependency Distance (MDD) than German sentences (Eppler, 2013). This is mainly due to the freer word order in German (Eppler, 2013). A smaller MDD means that two syntactically related words in German tend to stay together in a sentence. Therefore, it would be easier to learn more rules from German dependency trees in a length limitation. In addition, these rules tend to cover related syntactic components in German because they are more likely to be together.

- ZH–EN contains more long-distance word reordering than DE–EN. Based on the fact that the distortion function d in the phrase-based model measures the reordering distance, in our experiments we found that the distortion value averaged on phrases is 1.25 on ZH–EN and 0.18 on DE–EN. This suggests that our model requires a stronger capability to perform phrase reordering on ZH–EN.

2.4.1 Data Sets

The ZH–EN training corpus is from the LDC data, including LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08, and LDC2005T06. NIST 2002 (MT02) is taken as a development set to tune weights while NIST 2004 (MT04) and NIST 2005 (MT05) are used as the test data to evaluate systems. The Stanford Chinese word segmenter (Chang et al., 2008) is used to segment Chinese sentences into words. The Stanford dependency parser (Chang et al., 2009) parses a Chinese sentence into a projective dependency tree.[2] Table 2.4 provides a summary of the ZH–EN corpus.

ZH–EN	#Sentences	#Words (ZH)	#Words (EN)
Train	1,501,652	38,388,118	44,901,788
MT02	878	22,655	26,905
MT04	1,597	43,719	52,705
MT05	1,082	29,880	35,326

Table 2.4: Chinese–English corpus. For the English dev (MT02) and test (MT04 and MT05) sets, word counts are averaged across 4 references.

[2] In projective dependency trees, each subtree covers a continuous span of an input sentence.

DE–EN	#Sentences	#Words(DE)	#Words(EN)
Train	2,037,209	52,671,991	55,023,999
WMT11	3,003	72,661	74,753
WMT12	3,003	72,603	72,988
WMT13	3,000	63,412	64,810

Table 2.5: German–English corpus. In the dev (WMT11) and test (WMT12 and WMT13) sets, there is only one English reference for each German sentence.

The DE–EN training corpus is from WMT 2014, including Europarl V7 and News Commentary. News-Test 2011 (WMT11) is taken as a development set while News-Test 2012 (WMT12) and News-Test 2013 (WMT13) are our test sets. We tokenize German sentences with scripts in Moses (Koehn et al., 2007) and use mate-tools[3] to perform morphological analysis and parse the sentences (Bohnet, 2010). Then, MaltParser[4] converts the parse results into projective dependency trees (Nivre and Nilsson, 2005). Table 2.5 provides a summary of the DE–EN corpus.

2.4.2 Settings

For both language pairs, we filter sentence pairs longer than 80 words and keep the length ratio less than or equal to 3. English sentences are tokenized by scripts in Moses. Word alignment is performed by GIZA++ (Och and Ney, 2003) with the heuristic function *grow-diag-final-and* (Koehn et al., 2003). We use SRILM (Stolcke, 2002) to train a 5-gram language model on the Xinhua portion of the English Gigaword corpus 5th edition with modified Kneser-Ney discounting (Chen and Goodman, 1996). Batch MIRA (Cherry and Foster, 2012) is used to tune weights with a maximum iteration of 25. After tuning systems, we use feature weights which maximize BLEU (Papineni et al., 2002) scores on development sets to translate test sets. In each experiment, MIRA is run three times so that scores based on average performance and significance tests are more robust (Clark et al., 2011).

Automatic evaluation metrics are used to estimate translation performance of an SMT system by comparing system translations (hypotheses) with manual translations (references).

[3] `http://code.google.com/p/mate-tools/`
[4] `http://www.maltparser.org/`

In this thesis, we report scores from three widely used metrics: BLEU, METEOR (Denkowski and Lavie, 2011), and TER (Snover et al., 2006). While BLEU and METEOR are accuracy-based (the higher the better), TER is an error metric (the lower the better).

2.4.3 Baseline Systems

In this thesis, all systems are built using Moses which is a well-known translation framework and implements various models and algorithms of SMT. We build two baseline systems using Moses which are representatives of sequence-based and tree-based models, respectively:

PBMT is a phrase-based system with default configurations. By default, the maximum phrase length is set to 7. The system uses 8 standard features. In addition to an n-gram language model $p(t)$ over a translation t and a distortion function d for distance-based reordering, other features can be computed by summing over phrase pairs $\langle \overline{s}, \overline{t} \rangle$:

- translation probabilities $p(\overline{s}|\overline{t})$ and $p(\overline{t}|\overline{s})$ based on phrase counts
- lexical translation probabilities $p_{lex}(\overline{s}|\overline{t})$ and $p_{lex}(\overline{t}|\overline{s})$ (Koehn et al., 2003)
- phrase penalty $exp(-1)$
- word penalty $exp(|\overline{t}|)$

HPBMT is an HPB system with default configurations. By default, the maximum number of non-terminals is 2. During decoding, the maximum span of a non-glue rule is set to 20. We use 8 standard features. In addition to an n-gram language model, other features can be calculated by summing over translation rules $\langle \gamma, \alpha, \sim \rangle$.

- translation probabilities $p(\gamma|\alpha)$ and $p(\alpha|\gamma)$
- lexical translation probabilities $p_{lex}(\gamma|\alpha)$ and $p_{lex}(\alpha|\gamma)$
- rule penalty $exp(-1)$
- word penalty $exp(|\alpha|)$
- glue rule penalty $exp(-1)$ when a glue rule is used

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	PBMT	33.2	31.8	19.5	21.9
	HPBMT	36.5*	34.3*	20.5*	23.0*
METEOR ↑	PBMT	32.1	32.4	28.0	29.2
	HPBMT	33.0*	33.2*	28.5*	29.8*
TER ↓	PBMT	60.7	61.6	63.7	60.2
	HPBMT	59.2*	60.4*	62.8*	59.3*

Table 2.6: Evaluation results of PBMT and HPBMT on ZH–EN and DE–EN. $*$ means HPBMT is significantly better than PBMT at $p \leq 0.01$.

Table 2.6 shows evaluation scores of the two systems. We found that HPBMT is consistently better than PBMT on both language pairs. This is not surprising since HPBMT is a tree-based system which allows long-distance reordering and generalizations. In addition, in theory, rules in PBMT are a subset of rules in HPBMT.

2.5 Summary

In this chapter, we described statistical models in three groups: sequence-based models, tree-based models, and graph-based models. We listed the shortcomings of sequence-based and tree-based models and hypothesize that graphs-based models can make use of merits of both and alleviate their disadvantages. Furthermore, we explained the experimental set-up in this thesis and built two representative baseline systems using Moses: a phrase-based system and an HPB system.

In the next chapter, we will present an improved dependency tree-to-string model which allows non-syntactic phrases by using dependency decomposition. The improved dependency tree-to-string model shows the potential of graph-based models and motivates us to design graphs to combine trees and sequences.

The proper method for inquiring after the properties
of things is to deduce them from experiments.
— Isaac Newton

Chapter 3

Improved Dependency Tree-to-String Translation

As described in Chapter 2, both sequence-based and tree-based models have been widely explored in the field of SMT. While sequence-based models make use of local and sequential information, tree-based models take long-distance relations into consideration. In this sense, they complement each other. Therefore, an obvious thing to try is to combine them together in an attempt to obtain a better model.

In this chapter, we present such a combination which improves a Dependency Tree-to-String (Dep2Str) model (Xie et al., 2011) by encouraging non-syntactic phrases to be used. In the rest of this chapter, we (i) first review the Dep2Str model proposed by Xie et al. (2011) (Section 3.1), which is also one of the baselines of the whole thesis, and its advantages and disadvantages; (ii) provide a simple re-implementation of the Dep2Str model in Moses (Koehn et al., 2007) by converting dependency trees into constituent-style trees; (iii) propose an improved Dep2Str model by decomposing a dependency structure into smaller parts (Section 3.3). During decoding, such decomposition is enabled by using pseudo forests. Therefore, our model is also called a **pseudo forest-to-string** model. *Pseudo* means that the

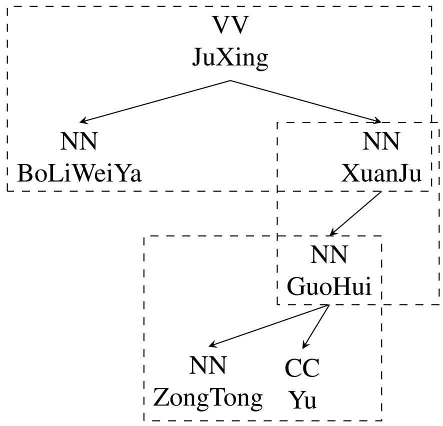


Figure 3.1: Illustration of head-dependent fragments (in rectangles) in a dependency tree of a Chinese sentence *BoLiWeiYa JuXing ZongTong Yu GuoHui XuanJu*. In this figure, each node is composed of two elements: a word and its POS tag.

forest is not obtained by combining several trees from a parser, but rather that it is created based on the decomposition of a dependency structure. Experimental results are presented in Section 3.4, including the integration of phrasal rules which are directly implied by bigram relations to further improve our system.

3.1 Introduction to Dependency Tree-to-String Model

In recent years, dependency trees have been widely used in SMT as they directly provide semantic relations between words and have the best inter-lingual phrasal cohesion property, i.e. phrases in one language tend to stay together during translation (Fox, 2002). Xie et al. (2011) presented a simple yet effective Dep2Str model. Different from treelet-based models which did not use synchronous grammars (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007), this model learns SCFG-like rules based on one-level subtrees. With the help of non-terminal mappings in rules, the model generates a complete translation by directly substituting non-terminals with previous hypotheses while at the same time reordering these hypotheses.

The one-level subtrees are called Head-Dependent (HD) fragments which are the basic translation units. Figure 3.1 shows examples of HD fragments. In a dependency tree, each non-leaf node is a head of some other nodes (dependents). An HD fragment is composed of a head node and all of its dependents. We can easily find that an HD fragment can be used to generate a dependency tree by recursively replacing its dependents with other HD fragments. Such a recursion implies a translation process: a dependency tree can be translated by translating its HD fragments in post-order and recursively combining their translations.

3.1.1 Examples of Rules

The Dep2Str model extracts two kinds of translation rules:

head rules which specify translations of source words. For instance, the following is a head rule to translate a Chinese word into an English word:

$$\text{JuXing} \rightarrow \text{holds}$$

HD rules each of which consists of three parts: an HD fragment s on the source side, a target string t, and a one-to-one mapping ϕ between non-terminals in s and t. For example:

$$\begin{aligned} s &= \text{(BoLiWeiYa) JuXing } (x_1\text{:XuanJu}) \\ t &= \text{Bolivia holds } x_1 \\ \phi &= \{x_1\text{:XuanJu} \rightarrow x_1\} \end{aligned}$$

is an HD rule where x_i are non-terminals which are constrained either by words (such as x_1:XuanJu in this example) or POS tags (such as x_1:NN). The source HD fragment is represented in a bracketed representation where symbols in brackets are dependents (such as *BoLiWeiYa*), while a symbol outside the brackets is the head node (for instance, *JuXing*).

3.1.2 Rule Extraction

Given a source dependency tree, a target string, and the word alignment between the source and target, the model firstly annotates each node n with two annotations:

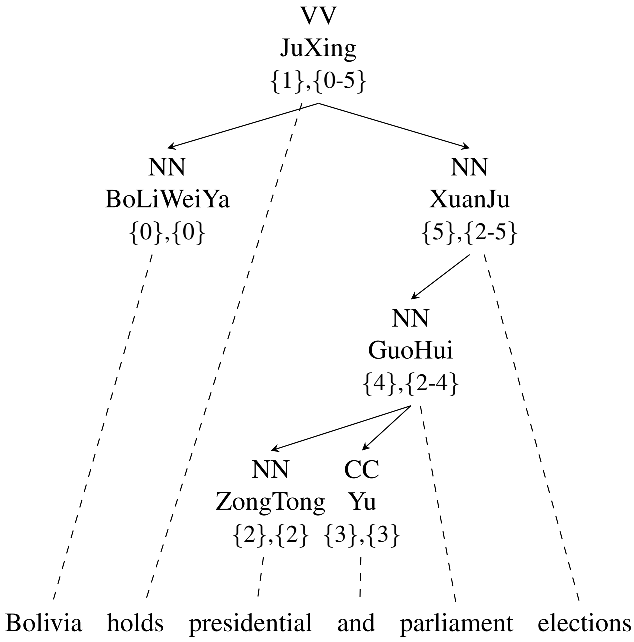


Figure 3.2: Illustration of annotations of head spans (the first {}) and dependency spans (the second {}) on each node. While a head span of a node n is a target span aligned to n, its dependency span is a target span consisting of head spans of all consistently word-aligned descendants of n. Dashed lines are word alignments.

head span is a minimal set of contiguous target positions (called closure) which are aligned to the node n.

dependency span is a minimal closure consisting of head spans of nodes, which are consistent with the word alignment, in the subtree rooted at n.

Figure 3.2 shows an example of annotating a dependency tree. By definition, the two spans specify the corresponding target positions of nodes (by head spans) and subtrees (by dependency spans), respectively.

After the annotation, head rules can be easily extracted on each node when its head span is consistent with the word alignment. HD rules need to be induced on acceptable HD fragments, where the head span of the head node is word-alignment consistent and none of dependency spans of its dependents is empty. We can find that the head span of the head node and dependency spans of its dependents in an acceptable HD fragment do not overlap with each other. This property makes it easier to extract HD rules.

Lexicalized HD rules (for instance, the example shown in Section 3.1.1) are firstly extracted where the head node and leaf nodes are represented by words, while the internal nodes are replaced by non-terminals which are constrained by words. The target side corresponding to an acceptable HD fragment and the mapping between non-terminals are determined by the head span of the head node and the dependency spans of its dependents. Then, lexicalized rules are generalized to unlexicalized HD rules by replacing nodes with non-terminals constrained by POS tags. For example:

$$\begin{aligned} s &= (x_1\text{:NN})\ \text{JuXing}\ (x_2\text{:NN}) \\ t &= x_1\ \text{holds}\ x_2 \\ \phi &= \{x_1\text{:NN} \rightarrow x_1,\ x_2\text{:NN} \rightarrow x_2\} \end{aligned}$$

is an unlexicalized HD rule.

3.1.3 Decoding

Translations are generated by applying rules to an input dependency tree using the chart decoder based on the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970). Figure 3.3 shows a derivation for translating a Chinese dependency tree into an English string in the Dep2Str model. The derivation proceeds from bottom to top. Each time either a source word or an HD fragment is translated. Non-terminals in HD rules are recursively substituted by previous hypotheses.

Similar to the HPB model (Chiang, 2005; Chiang, 2007), the final translation is obtained by finding the best derivation d^* from all possible derivations D which convert the source dependency into target strings, as in Equation (3.1):

$$d^* = \operatorname*{argmax}_{d \in D} p(d) \approx \operatorname*{argmax}_{d \in D} \prod_i \phi_i(d)^{\lambda_i} \tag{3.1}$$

where $\phi_i(d)$ are feature functions defined on a derivation d, and λ_i are feature weights.

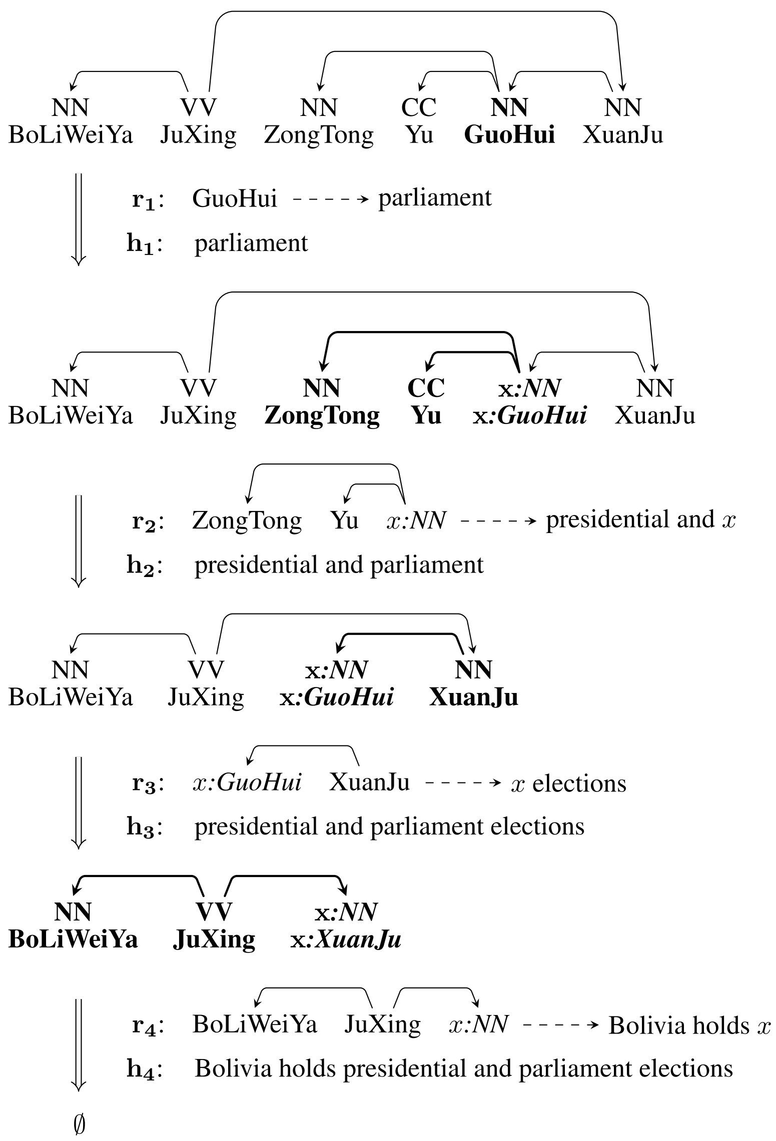


Figure 3.3: A derivation of translating a Chinese dependency tree into an English string in the Dep2Str model. r_i are rules, each of which covers a subtree (in bold). h_i are hypotheses. x are non-terminals which are constrained by either words (e.g. x:*GuoHui*) or POS tags (e.g. x:*NN*).

3.2 Dependency Transformation

Before improving the Dep2Str model, we first implement the model in Moses. Recall that the conventional chart decoder in Moses is used to translate sentences or constituent trees based on SCFG rules. Each time a subtree covering a continuous span of an input sentence is translated. The decoder can also be used to translate dependency trees in the Dep2Str model because Dep2Str rules are SCFG-like, i.e. HD fragments are flat (one-level subtree) and each rule covers a sequence of continuous words.

However, one difference is that HD rules have encoded dependency information, namely indications of head nodes and dependents. In addition, dependency trees and constituent trees are in different representations: while in a constituent tree input words are leaf nodes and all non-leaf nodes are labeled by categories which are directly used as non-terminals in a tree-based model, in a dependency tree each node is labeled by an input word and thus non-terminals are not explicitly represented.

Therefore, to reuse the chart decoder in Moses without making any changes, in this section we introduce an algorithm which transforms a dependency tree into a corresponding constituent-style tree where each internal node is labeled by non-terminals defined in the Dep2Str model associated with dependency information.

Note that this conversion is performed only on an input dependency tree for the decoding step. This means during training we still use dependency trees. In addition, the transformation is different from other work which transforms a dependency tree into a constituent tree (Collins et al., 1999; Xia and Palmer, 2001). In this chapter, the produced constituent tree preserves dependency relations between words and is directly derived from the dependency structure without refinement. Accordingly, the constituent tree may not be in a linguistically well-formed syntactic structure. However, it is not a problem to our model, because we only need the dependency information.

Algorithm 3.1: Algorithm for transforming a dependency tree to a constituent tree.

```
1  Function TransformDep(h_d, h_c)
2      n_c ← h_d.pos + H                              // e.g. NN:H (head)
3      add n_c to h_c.kids
4      add h_d.word to n_c.kids                       // leaf node e.g. GuoHui
5      for n_d ∈ h_d.kids do
6          n_c ← n_d.pos + n_d.dep  // e.g. NN:L2 (the second left dependent)
7          add n_c to h_c.kids
8          if n_d is leaf then
9              add n_d.word to n_c.kids
10         else
11             n_w ← n_d.word + n_d.dep
12             add n_w to n_c.kids
13             TransformDep(n_d, n_w)
14         end
15     end
16 end
```

3.2.1 Algorithm

The transformation is executed by a function `TransformDep(`$T_d.root, T_c.root$`)`, where T_d is a dependency tree, T_c is a resulting constituent tree. $T_c.root$ is initialized as S. Algorithm 3.1 shows a pseudo code for the function, which proceeds recursively from top to bottom on each HD fragment in a dependency tree. For a current HD fragment rooted at h_d, the corresponding constituent subtree is rooted at h_c. We first create an internal constituent node n_c which is a child of h_c and labeled by the combination (indicated by $+$) of the POS tag $h_d.pos$ of h_d and its dependency information H (Line 2). Then, a leaf node labeled by the word $h_d.word$ is created and added to the constituent tree as a kid of n_c (Line 3). After processing the current node h_d, we traverse dependents of h_d. For each dependent n_d, we create a new internal node n_c and add it to $h_c.kids$ (Lines 6–7). If n_d is a leaf node, we create a constituent node labeled by $n_d.word$ and add it to the constituent tree as a leaf node (Lines 8–9). Otherwise, we create a new internal node n_w which is labeled by the combination of the word $n_d.word$ and a dependency information $n_d.dep$ (Lines 11–12). Finally, we recursively call `TransformDep(`n_d, n_w`)` to process the next HD fragment rooted as n_d (Line 13).

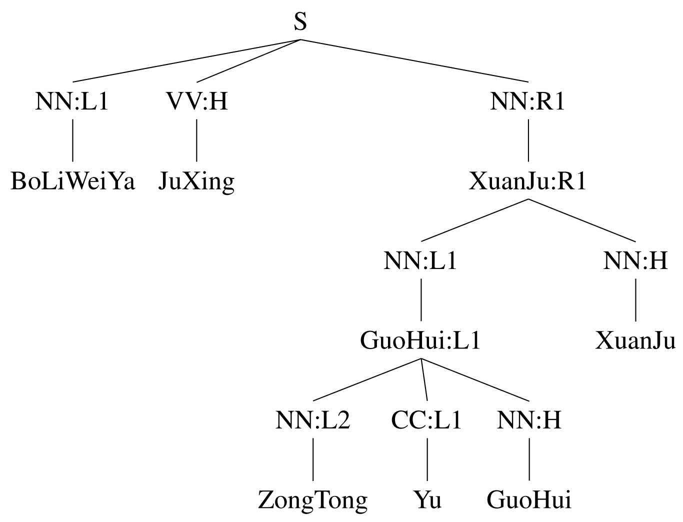


Figure 3.4: Constituent-style tree converted from the dependency tree in Figure 3.1. Each internal node is constrained by dependency information, such as *L1* means the first left dependent, *R1* means the first right dependent, and *H* means a head node.

We can find that for the leaf nodes and head node in an HD fragment, we create constituent nodes each of which only covers one word. For an internal node n, we create constituent nodes which cover all the words in the subtree rooted at n. Each internal node in the constituent tree is constrained by dependency information. Taking the dependency tree in Figure 3.1 as an example, the transformation result is shown in Figure 3.4. In the Dep2Str model, a leaf node can be replaced by a non-terminal constrained by its POS tag, so for the leaf node *ZongTong* in the HD fragment *(ZongTong) (Yu) GuoHui*, we create a constituent node *NN:L2*, where *NN* is the POS tag while *L2* denotes that the leaf node *ZongTong* is the *second left* dependent of the head node *GuoHui*. For the internal node *GuoHui* in the HD fragment *(HuoHui) XuanJu*, we create two constituent nodes which cover all words in the dependency subtree rooted at this node, with one of them labeled by the word itself. Both nodes are constrained by dependency information *L1*. After such a transformation is conducted on each HD fragment recursively, we obtain the constituent tree.

3.2.2 Evaluation

This transformation makes our implementation of the Dep2Str model easier, because we can use the tree-to-string decoder in Moses. All we need to do is to write a new rule extractor which extracts head rules and HD rules and represents these rules in the format defined in Moses. Taking the rule in Section 3.1.1 as an example, its representation in Moses is:

$$\begin{aligned} s &= \text{BoLiWeiYa JuXing [XuanJu:R1][X] [H1]} \\ t &= \text{Bolivia holds [XuanJu:R1][X] [X]} \\ \phi &= \{2 \rightarrow 2\} \end{aligned}$$

where *H1*[1] denotes the position of the head word as 1, *R1* indicates the first right dependent of the head word, *X* is a general label on the target side, and ϕ is the set of mappings between non-terminals in s and t.[2]

The Dep2Str system we implemented in Moses is denoted as **D2S**. The first experiment we conduct is to sanity check the implementation. We take the system in Xie et al. (2011) (denoted as **XJ**) as a comparison. Both systems are trained and tested on the same data as in Xie et al. (2011). BLEU scores of both systems are as follows:

System	MT05
XJ	33.91
D2S	33.79

We found that, using the transformation of dependency trees, the Dep2Str model implemented in Moses (D2S) is slightly worse than the standard implementation (XJ). But the small under-performance is not significant.

3.3 Dependency Decomposition

The Dep2Str model treats whole HD fragments as the basic units which only cover syntactic phrases. Because dependency structures are flatter than constituent trees (Meng et al., 2013;

[1] In experiments, during decoding the matching of the H1 is ignored.

[2] More detail on the rule representation in Moses can be found at `http://www.statmt.org/moses/?n=Moses.SyntaxTutorial`.

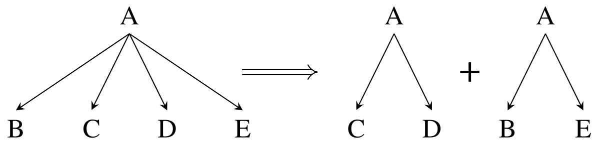


Figure 3.5: Illustration of decomposing an HD fragment over a sequence *B C A D E* into two parts: one of them covers *C A D* while another one covers *B A E*. $+$ means the large fragment can be obtained by attaching dependents of the right sub-fragment to the left sub-fragment.

Xie et al., 2014), this may result in a data-sparsity problem and thus a lower phrase coverage. Therefore, in this section we describe a decomposition method to make use of fragments which cover non-syntactic phrases. For example, as shown in Figure 3.5, after decomposition we can use the two fragments on the right which are non-syntactic phrases.

We assume that a large HD fragment is formed by attaching (indicated by $+$) dependents to a small HD fragment. For simplicity, such an attachment is carried out in one step. This means that an HD fragment is decomposed into two smaller parts, which is formulated in Equation (3.2):

$$\begin{aligned} L_i \cdots L_1 H R_1 \cdots R_j &= L_m \cdots L_1 H R_1 \cdots R_n \\ &\quad + L_i \cdots L_{m+1} H R_{n+1} \cdots R_j \\ \text{subject to} & \\ & m + n > 0, i \geq m, j \geq n \end{aligned} \tag{3.2}$$

where H denotes the head node, L_i are left dependents, and R_j are right dependents. m and n are two decomposition positions. Equation (3.2) indicates that, given an HD fragment $L_m \cdots L_1 H R_1 \cdots R_n$, we can obtain a larger fragment by adding (indicated by $+$) dependents $L_i \cdots L_{m+1}$ to its left and $R_{n+1} \cdots R_j$ to its right. Let us call $L_m \cdots L_1 H R_1 \cdots R_n$ a *core* fragment while $L_i \cdots L_{m+1} H R_{n+1} \cdots R_j$ is a *shell* fragment. To better understand the decomposition, Figure 3.5 shows an example of decomposing an HD fragment *(B) (C) A (D) (E)* into two smaller fragments: a *core (C) A (D)* and a *shell (B) A (E)*.

The decomposition enables us to build a model which extracts translation rules from sub-fragments to enrich the rule set and create a pseudo forest from an input dependency

Algorithm 3.2: Algorithm for extracting rules on decomposed HD fragments.

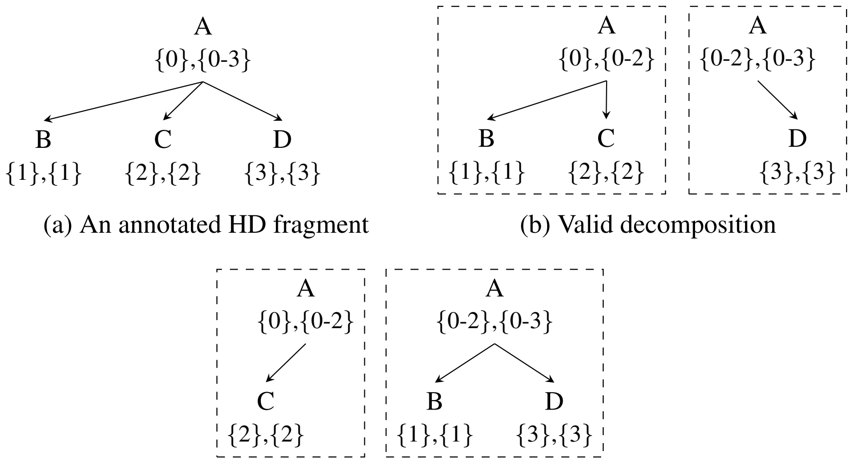
Data: annotated HD fragment F
Result: a set of sub-fragment rules R
1 $R \leftarrow \{\}$
2 **forall** $\langle F_c, F_s \rangle$, $F_c + F_s = F$ **do**
3   update annotations on F_c and F_s
4     $d_c \leftarrow F_c.head.dspan$
5     $D_s \leftarrow \{d.dspan, \forall d \in F_s.dependents\}$
6   **if** $\forall d_s \in D_s, d_c \cap d_s = \emptyset$ **then**
7   add HD rules extracted on F_c to R
8         $F_s.head.hspan \leftarrow d_c$
9   set $F_s.h$ unlexicalized
10   add HD rules extracted on F_s to R
11   **end**
12 **end**

tree to allow for translating an HD fragment in two steps. Note that different values of m, n in Equation (3.2) result in different decompositions. Although ideally only reliable decompositions should be allowed, it is non-trivial to measure the reliability. Therefore, for simplicity, in this chapter we take all possible decompositions into consideration and let the decoder decide which one should be used during decoding.

3.3.1 Rule Extraction

In the Dep2Str model, rules are extracted on complete HD fragments. When the decomposition is allowed, we also extract sub-fragment rules by taking each sub-fragment as a new HD fragment. Algorithm 3.2 shows an extraction process. In Algorithm 3.2, we go through possible decompositions $\langle F_c, F_s \rangle$ of an HD fragment F, each of which consists of two sub-fragments: *core* F_c and *shell* F_s (Line 2) which are treated as new HD fragments and new annotations are assigned (Line 3). Then, HD rules are extracted as described in Section 3.1.2. In this section, we use $F.head$ and $F.dependents$ to denote the head node and dependents in F. $n.hspan$ and $n.dspan$ represents the head span and dependency span of a node n.

During training, not all decompositions are used to extract rules. Given a word alignment,



(a) An annotated HD fragment

(b) Valid decomposition

(c) Invalid decomposition

Figure 3.6: Illustration of valid and invalid decompositions during training. (a) is an annotated HD fragment, (b) is a valid decomposition, while (c) is an invalid decomposition because the dependent span $\{0\text{-}2\}$ of *A* in the left fragment overlaps with the dependent span $\{1\}$ of *B* in the right fragment. In (b) and (c), left fragments are *cores* while right ones are *shells*.

we only focus on decompositions where the F_c does not overlap with F_s in terms of their aligned target spans (Lines 4–6). Figure 3.6 shows examples of valid and invalid decompositions. The decomposition in Figure 3.6c is invalid because the dependency span $\{0\text{-}2\}$ of the node A in the left fragment overlaps with the dependency span $\{1\}$ of the node B in the right fragment. When we extract HD rules on F_s, the head span of its head node is set to the dependency span of the head node in F_c (Line 8). For example, the node A in the *shell* in Figure 3.6b has a head span $\{0\text{-}2\}$. In addition, the head node in F_s is always unlexicalized (Line 9) when we induce HD rules.

Note that different from other work which combines non-syntactic phrases into the Dep2Str model by using special labels as described in Section 2.2.3, our sub-fragment rules are standard HD rules, which can be directly used without modifying the decoder.

3.3.2 Creating Pseudo Forests

The decomposition allows us to translate a large HD fragment in two steps. Recall that the sub-fragment *core* covers a continuous phrase of a source sentence. Accordingly, we

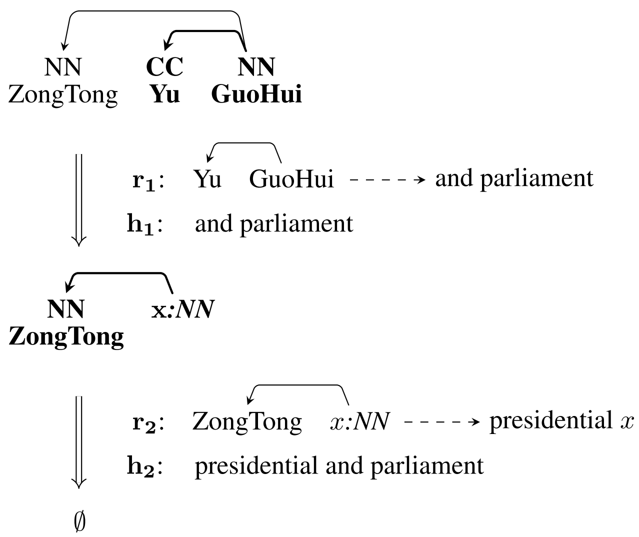


Figure 3.7: An example of translating a large HD fragment using dependency decomposition in two steps. r_i are rules which are applied to sub-structures in bold. h_i are hypotheses.

translate it first and then build a whole translation by translating another sub-fragment *shell*. Figure 3.7 shows a decomposed translation step, where the sub-fragment *(Yu) GuoHui* is firstly translated and its translation is subsequently combined with the translation of another sub-fragment *(ZongTong) GuoHui* by treating the head node *GuoHui* as a non-terminal.

To allow decomposition during decoding, instead of taking a dependency tree as an input and looking for rules to translate sub-fragments, we directly encode the decomposition into the input dependency tree with the result being a *pseudo* forest. Based on the transformation algorithm in Section 3.2, the pseudo-forest can also be represented in a constituent-tree style and decoded without making changes to the decoder. Different from forests used in forest-based models (Mi et al., 2008; Tu et al., 2010), such a pseudo forest is not aimed at efficiently reducing the influence of parsing errors, but it is easily available and compatible with the Dep2Str Model.

Figure 3.8 shows an example of a pseudo forest created from the dependency tree in Figure 3.1. According to the definition of the decomposition, we only create a forest structure for each HD fragment. For example, based on the decomposition in Figure 3.7, we create a constituent node labeled with *NN:H* that covers the sub-fragment *(Yu) Guohui*. Accordingly,

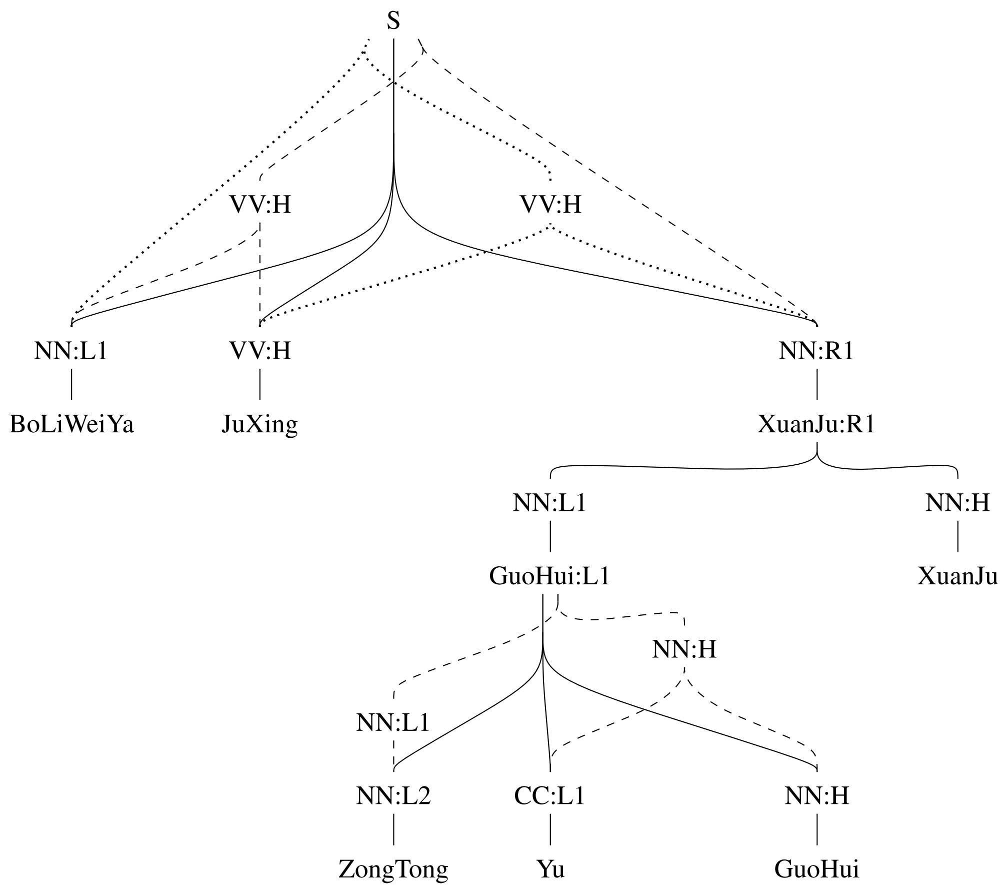


Figure 3.8: An example of a pseudo forest created for the dependency tree in Figure 3.1 using the constituent style (as in Figure 3.4) described in Section 3.2. Dashed and dotted edges are introduced by dependency decomposition.

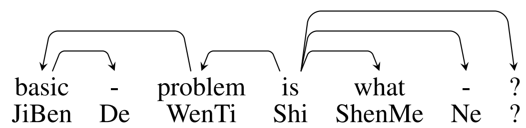
a new node labeled with *NN:L1* is also created, which covers the node *ZongTong*, because it is now the first left dependent in the sub-fragment *(ZongTong) GuoHui*.

3.4 Experiments

We conduct experiments on the ZH–EN and DE–EN language pairs. All our systems are implemented in Moses with the standard features as used in the HPBMT in Chapter 2. We first compare D2S with HPBMT and PBMT to explore its advantages and disadvantages. Then, we present results to investigate the influence of dependency decomposition, followed by experiments on integrating phrasal rules.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	PBMT	33.2	31.8	19.5	21.9
	HPBMT	36.5	34.3	20.5	23.0
	D2S	35.1*	33.1*	20.0*	22.3*
METEOR ↑	PBMT	32.1	32.4	28.0	29.2
	HPBMT	33.0	33.2	28.5	29.8
	D2S	31.5	31.4	28.2*	29.5*
TER ↓	PBMT	60.7	61.6	63.7	60.2
	HPBMT	59.2	60.4	62.8	59.3
	D2S	58.1*	59.6*	63.4*	60.1

Table 3.1: Evaluation results of D2S, which re-implements Xie et al. (2011), compared with PBMT and HPBMT. $*$ means D2S is significantly better than PBMT at $p \leq 0.01$.

Source:


Ref: What are the basic problems ?

PBMT: The basic question is what ?

D2S: What is the basic problem ? ?

Figure 3.9: Examples of translations generated by D2S and PBMT from Chinese to English. Chinese words are accompanied by their English meaning. Compared with PBMT, D2S generated a better translation.

3.4.1 Basic Results

We first compare the baseline D2S with PBMT and HPBMT. Table 3.1 shows evaluation scores calculated by three metrics on the two language pairs. We found that, in terms of BLEU and TER, the D2S system is significantly better than PBMT. For example, in terms of BLEU, the improvements are +1.6 (4.9%, relative) on ZH–EN and +0.5 (2.2%, relative) on DE–EN. The improvement is reasonable since D2S is based on dependency structures which are linguistically informed and provide long-distance relations between words. Figure 3.9 shows examples of translations from D2S and PBMT, where D2S generated a better target sentence with a correct word order.

However, such an improvement is not consistent across different metrics as on ZH–EN PBMT has higher METEOR scores (+0.8 on average) than D2S. We observed that D2S

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	HPBMT	36.5	34.3	20.5	23.0
	D2S	35.1	33.1	20.0	22.3
	+Decomp	36.6*	34.9*	20.4*	22.7*
METEOR ↑	HPBMT	33.0	33.2	28.5	29.8
	D2S	31.5	31.4	28.2	29.5
	+Decomp	32.1*	32.1*	28.4*	29.7*
TER ↓	HPBMT	59.2	60.4	62.8	59.3
	D2S	58.1	59.6	63.4	60.1
	+Decomp	57.5*	58.7*	63.4	60.0

Table 3.2: Evaluation results of the decomposition approach (+Decomp) compared with D2S and HPBMT. $*$ means the decomposition method significantly improves D2S at $p \leq 0.01$.

produces significantly shorter translations than that of PBMT on ZH–EN, which might cause the lower METEOR scores as METEOR favors longer sentences (He and Way, 2010). In addition, compared with HPBMT, D2S performs significantly worse: -1.3 (3.7%, relative) BLEU on ZH–EN and -0.6 (2.8%, relative) BLEU on DE–EN. We presume the main reason is that D2S has a severe problem with the coverage of non-syntactic phrases because of the flatness of dependency trees (Meng et al., 2013; Xie et al., 2014). Experiments in the next sections will show that taking non-syntactic phrases into consideration significantly improves translation performance of D2S as observed in other work (Koehn et al., 2003; Xie et al., 2014).

3.4.2 Influence of Decomposition

We further examine the effectiveness of the decomposition approach by adding it to D2S (denoted by +Decomp). Table 3.2 shows evaluation results. We found that decomposition consistently improves the baseline system D2S on both language pairs and results in a system only slightly worse than HPBMT. The improvement is based on the fact that decomposition provides more translation rules and allows for the translation of large HD fragments by combining translations of its smaller parts. Figure 3.10 shows examples of translations where the D2S+Decomp successfully translated a Chinese word *HouNian* into *the year of the monkey* by using a new rule which translated the sub-fragment *(Nian) Shi (HouNian)*.

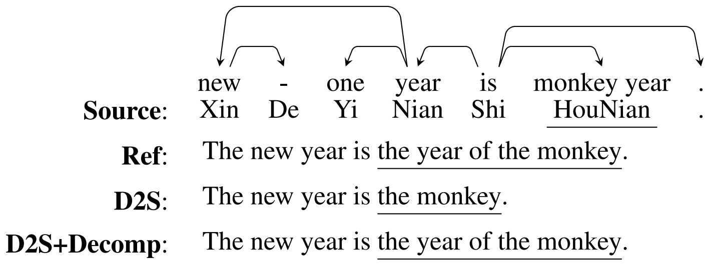


Figure 3.10: Examples of translations from D2S with the decomposition approach (+Decomp). D2S+Decomp generated a better translation than D2S for the underlined Chinese word.

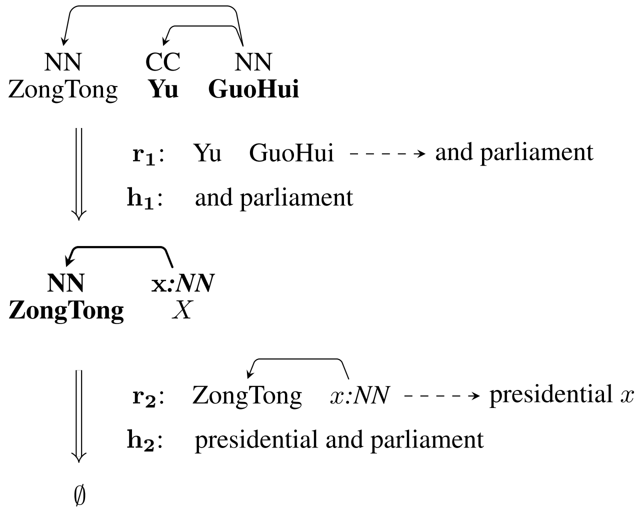


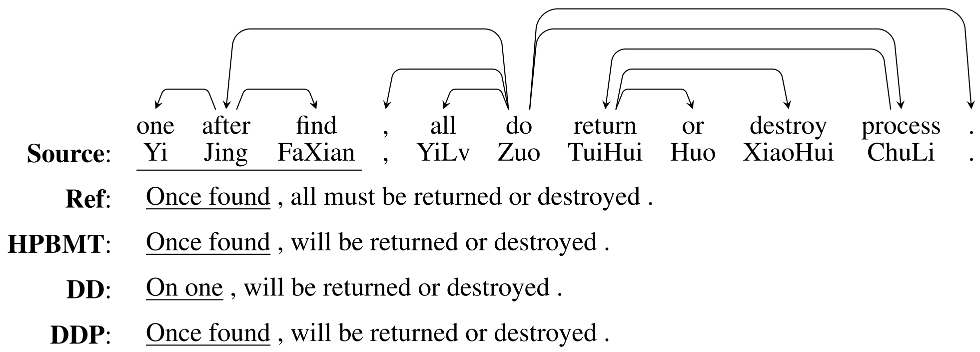
Figure 3.11: An example of translating an HD fragment using a phrasal rule r_1. X is a general non-terminal for integrating phrasal rules in Moses. Note that phrasal rules are fully lexicalized and do not contain structural information.

3.4.3 Integrating Phrasal Rules

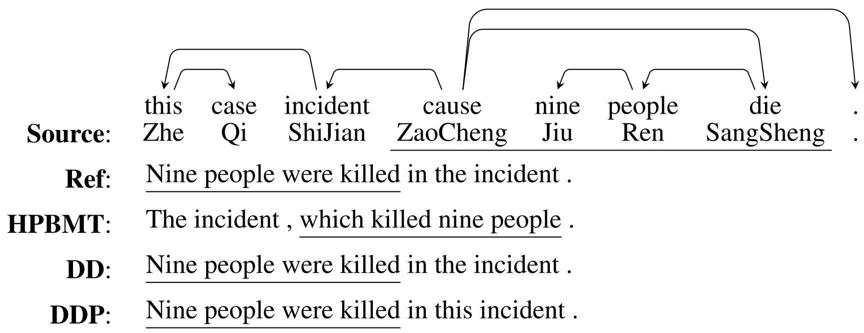
As described in Section 3.1, the Dep2Str model only extracts phrasal rules for translating source words (head rules). Although the decomposition allows the model to cover some non-syntactic phrases, the model could be further enhanced by including phrasal rules that cover multiple source words and are purely lexicalized without any dependency information included. In this section, we conduct experiments where phrase pairs from PBMT are added into our system resulting in a so-called *mixture syntax-based system* (Hoang, 2011). These phrasal rules can be directly handled by Moses which creates a general non-terminal X for

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	HPBMT	36.5	34.3	20.5	23.0
	D2S+Decomp	36.6	34.9	20.4	22.7
	+Phrase	37.7^{*+}	35.5^{*+}	20.8^{*+}	23.4^{*+}
METEOR ↑	HPBMT	33.0	33.2	28.5	29.8
	D2S+Decomp	32.1	32.1	28.4	29.7
	+Phrase	32.8^{*}	32.8^{*}	28.7^{+*}	30.0^{*+}
TER ↓	HPBMT	59.2	60.4	62.8	59.3
	D2S+Decomp	57.5	58.7	63.4	60.0
	+Phrase	56.9^{*+}	58.0^{*+}	62.7^{*}	59.1^{*}

Table 3.3: Evaluation results of D2S when phrasal rules are integrated. $*$ means phrasal rules significantly improve D2S+Decomp at $p \leq 0.01$. $+$ means D2S+Decomp+Phrase is significantly better than HPBMT at $p \leq 0.01$.



(a) DDP, HPBMT vs DD



(b) DDP, DD vs HPBMT

Figure 3.12: Examples of translations from D2S with phrasal rules. DD represents D2S+Decomp, while DDP denotes D2S+Decomp+Phrase. (a) shows the usefulness of phrasal rules which translated a Chinese phrase (underlined) into a correct translation, while (b) illustrates the benefit of using dependency structures which helped to generate better translations with a correct word order.

each source span so that translations of phrases can be fully accessed. If a phrase is covered by a constituent node, its translations will be treated as translations of the subtree rooted at this node. Otherwise, it will be covered by glue rules which monotonically combine translations of each span. Figure 3.11 shows an example of translating an HD fragment using a phrasal rule which is covered by a complete subtree.

Table 3.3 shows evaluation results of systems after integrating phrasal rules. We found that phrasal rules significantly improve our system, e.g. +0.9 (2.4%, relative) BLEU on ZH–EN and +0.6 (2.6%, relative) BLEU on DE–EN. The final system is significantly better than HPBMT, e.g. +1.2 (3.4%, relative) BLEU on ZH–EN and +0.4 (1.6%, relative) BLEU on DE–EN. Figure 3.12a shows translation examples where phrasal rules correctly translated a Chinese phrase *Yi Jin FaXian* to English *once found* in both HPBMT and D2S+Decomp+Phrase. Figure 3.12b shows that, compared with HPBMT, systems based on dependency structures performed a better phrase reordering resulting in better translations.

3.4.4 Comparison of Model Size

Besides long-distance reordering (Xie et al., 2011), another attraction of using dependency structures is that it can significantly reduce the size of learned models. Table 3.4 shows the number of rules in different systems. It is easy to see that all of our systems use significantly fewer rules than HPBMT. However, using fewer rules does not mean worse translation performance. As shown in Table 3.3 and described in Section 3.4.3, our system D2S+Decomp+Phrase achieves significantly better translation results than HPBMT.

System	# Rules	
	ZH–EN	DE–EN
HPBMT	388M	684M
D2S	27M	41M
+Decomp	84M	92M
+Phrase	161M	206M

Table 3.4: The number of rules in HPBMT and D2S systems

3.5 Summary

In this chapter, we presented a pseudo forest-to-string model which improves the Dep2Str translation model by decomposing dependency structures so that non-syntactic phrases can be covered. To implement the model in Moses, we transformed an input dependency tree into a corresponding constituent-style tree before decoding which makes Moses perform dependency-based translation without necessitating any changes to the decoder. Furthermore, by resorting to phrasal rules, our system performed significantly better than the HPB model in Moses.

Experiments in this chapter showed that integrating non-syntactic phrases can significantly improve system performance. In the next chapter, we will present a graph-based model where the basic translation units are subgraphs, each of which covers either a syntactic phrase or a non-syntactic phrase without distinguishing them.

Divide each difficulty into as many parts as
is feasible and necessary to resolve it.
— René Descartes

Chapter 4

Non-recursive Graph-to-String Translation

In the previous chapter, we improved a dependency tree-to-string model (Xie et al., 2011) by allowing non-syntactic phrases to be covered. Experiments suggested that combining translation rules from trees and sequences is beneficial. However, the model treats trees and sequences as separate structures.

In this chapter, we construct graphs which combine trees and sequences in a unified representation and present a segmentational graph-based translation model (called **SegGBMT** in this thesis). The model segments an input graph into a sequence of subgraphs and generates a complete translation by combining translations of each subgraph. Our model is significantly different from previous work (shown in Table 4.1):

- The phrase-based model (Koehn et al., 2003) extends translation units from single words to continuous phrases. However, one of its significant weaknesses is that it cannot learn generalizations or translations of discontinuous phrases (any subset of words of an input sentence), such as French *ne...pas* to English *not* (Quirk et al., 2005; Galley and Manning, 2010).

Model	C	D	S
Koehn et al. (2003)	•		sequence
Menezes and Quirk (2005) and Quirk et al. (2005)		•	tree
Galley and Manning (2010)	•	•	sequence
Our model	•	•	graph

Table 4.1: Comparison between our SegGBMT model and existing similar work in terms of three aspects: keeping continuous phrases (C), allowing discontinuous phrases (D) and input structures (S).

- In treelet-based models (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007), the basic translation units are connected subgraphs of trees which may cover discontinuous phrases. However, continuous phrases which are not connected and thus excluded could in fact be extremely important to system performance (Koehn et al., 2003; Hanneman and Lavie, 2009).
- Although Galley and Manning (2010) directly make use of both continuous and discontinuous phrases without resorting to linguistic structures, plenty of unreliable phrases (Quirk et al., 2005) are extracted resulting in a huge model.

Different from previous work, in our model each subgraph covers either a continuous phrase (as in phrase-based models) or a discontinuous phrase (as in treelet-based models) without distinction. In the rest of this chapter, we firstly introduce how to construct graph structures (Section 4.1). Then, we describe our SegGBMT model which is based on graph segmentation (Section 4.2). The model is further improved by using a graph segmentation model which takes source context into consideration (Section 4.3). We explain experimental results in Section 4.4 followed by a short summary of this chapter (Section 4.5).

4.1 Dependency-Bigram Graphs

The graph used in this chapter directly combines a sequence and a dependency tree by using bigram links and dependency links. The graph is therefore called a Dependency-Bigram Graph (DBG). Formally, DBGs are connected, directed, and node-labeled. We do not

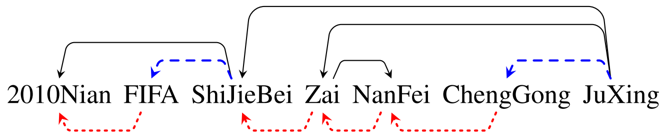


Figure 4.1: An example of a graph on a Chinese sentence. Dotted lines are bigram relations. Solid lines are dependency relations. Dashed lines are shared by bigram and dependency relations.

consider edge labels because they would complicate our explanations and did not bring significant improvement in our experiments (Section 4.4.6). Figure 4.1 shows an example of a DBG. Each edge in the graph denotes either a bigram relation or a dependency relation:

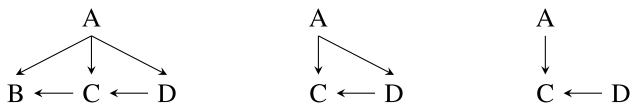
- Bigram relations are implied in sequences and provide local and sequential information on pairs of continuous words. Phrases connected by bigram relations (i.e. continuous phrases) are known to be useful for improving phrase coverage (Hanneman and Lavie, 2009).
- Dependency relations come from dependency structures which model syntactic and semantic relations between words. Phrases connected by dependency relations (i.e. treelets) are linguistically motivated and thus more reliable (Quirk et al., 2005).

By combining the two kinds of relations together in graphs, we can make use of both continuous and linguistically-informed discontinuous phrases as long as they are connected subgraphs. Based on a subset of nodes in a graph, a subgraph in this chapter is connected and includes all edges connecting those nodes (also called node-induced subgraph), as illustrated in Figure 4.2. [1]

In addition, our model can cover phrases which are unavailable in both the phrase-based model and treelet-based models, as shown in Figure 4.3. Given the graph in Figure 4.1, we can extract subgraphs covering three kinds of phrases:

- Phrases as in Figure 4.3a which are connected by bigram links: these phrases can be

[1] In this chapter, subgraphs are connected, and the terms *subgraph* and *node-induced subgraph* are used without distinction.



(a) (b) (c)

Figure 4.2: Illustration of node-induced subgraphs in this chapter. (a) is a graph and (b) is a subgraph of (a). But (c) is not considered as a valid subgraph because it lacks an edge from A to D.

(a) Subgraph connected by bigram links

(b) Subgraph connected by dependency links

(c) Subgraph connected by both dependency and bigram links

Figure 4.3: Examples of subgraphs extracted from Figure 4.1. Dotted lines are bigram relations. Solid lines are dependency relations.

covered by the phrase-based model rather than the treelet-based model when they are not connected by dependency links.

- Phrases as in Figure 4.3b which are connected by dependency links: these phrases can be extracted in the treelet-based model rather than the phrase-based model when they are discontinuous.

- Phrases as in Figure 4.3c which are not connected by a single type of links: however, since they are valid subgraphs in graph structures, we can make use of them as well. In experiments, we found 19.8%–24.4% of source subgraphs in our model are in this group.

4.2 Segmentation-Based Translation

Our SegGBMT model extends the phrase-based model by translating an input graph rather than a sequence to a target string. The graph is segmented into a sequence of connected and node-induced subgraphs $[G(\tilde{s}_{a_1}), \cdots, G(\tilde{s}_{a_I})]$ (called a graph segmentation), each of which is disjoint with others. Note that during segmenting a graph, nodes are divided into subgraphs and edges between subgraphs are ignored. Therefore, subgraphs in a graph

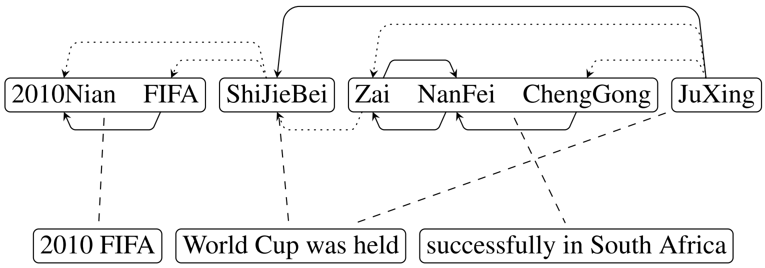


Figure 4.4: A source graph is segmented into three subgraphs each of which corresponds to a target phrase. Dashed lines denote alignments between source subgraphs and target phrases. Edges in dotted lines are ignored during segmenting the graph.

segmentation cover all nodes rather than edges. This also means that when subgraphs in a graph segmentation are combined to form a graph, their nodes remain disjoint and new edges are formed between them. Following the phrase-based model, our SegGBMT model is formulated in Equation (4.1):

$$\begin{aligned} p(G(\tilde{s}_1^I)|\bar{t}_1^I) &= \prod_{i=1}^{I} p(G(\tilde{s}_{a_i})|\bar{t}_i) d(G(\tilde{s}_{a_i}), G(\tilde{s}_{a_{i-1}})) \\ &\approx \prod_{i=1}^{I} p(G(\tilde{s}_{a_i})|\bar{t}_i) d(\tilde{s}_{a_i}, \tilde{s}_{a_{i-1}}) \end{aligned} \tag{4.1}$$

where $G(\tilde{s}_{a_i})$ denotes a node-induced source subgraph which covers a (discontinuous) phrase $\tilde{s}_i$ and corresponds to a continuous target phrase $\bar{t}_i$. Similar to the distortion function in the phrase-based model, d is a distortion model in SegGBMT which will be defined in Section 4.2.2. Figure 4.4 shows a graph–string pair where the graph is segmented into three subgraphs each of which is aligned to a target phrase.

4.2.1 Rule Extraction

Different from phrase-based models, the basic translation units in our SegGBMT model are subgraphs. Therefore, during training, we extract subgraph–phrase pairs $\langle G(\tilde{s}), \bar{t}\rangle$ instead of phrase pairs on a parallel graph–string sentence pair $\langle G(s), t, a\rangle$ associated with a word alignment a. An example of a translation rule is as follows:

2010Nian FIFA ShiJieBei $\dashrightarrow$ 2010 FIFA World Cup

Note that the source side of a rule in our model is a graph which is connected and does not contain non-terminals. The graph can be used to cover either a continuous phrase or a discontinuous phrase according to its match with an input graph during decoding. The target side of a rule is a continuous phrase.

The algorithm for extracting translation rules is shown in Algorithm 4.1, which is an extension of the phrase-pair extraction in phrase-based models. This algorithm traverses each phrase pair $\langle \tilde{s}, \bar{t} \rangle$, which is within a length limit L and consistent with a given word alignment a (lines 1–2), and outputs $\langle G(\tilde{s}), \bar{t} \rangle$ if $\tilde{s}$ is covered by a subgraph $G(\tilde{s})$ which is connected (lines 6–8). A source phrase can be extended with unaligned source words which are adjacent to the phrase (lines 9–14). We use a queue Q to store all phrases which are consistently aligned to the same target phrase (line 3).

Algorithm 4.1: Algorithm for extracting translation rules from a graph-string pair.

Data: graph–string pair $\langle G(s), t, a \rangle$
Result: translation pairs R
1 **forall** $\bar{t}$ in t: $\mid \bar{t} \mid \leq L$ **do**
2   find $\tilde{s}$ in S so that $\mid \tilde{s} \mid \leq L$ and $\langle \tilde{s}, \bar{t} \rangle$ is consistent with a
3     $Q \leftarrow \{\tilde{s}\}$
4   **while** $Q \neq \emptyset$ **do**
5         $\tilde{s} \leftarrow Q.pop()$
6   **if** $G(\tilde{s})$ is connected **then**
7   add $\langle G(\tilde{s}), t \rangle$ to R;
8   **end**
9   **if** $\mid \tilde{s} \mid < L$ **then**
10   **for** each unaligned word s_i adjacent to $\tilde{s}$ **do**
11                 $\tilde{s}' \leftarrow$ extend $\tilde{s}$ with s_i
12   add $\tilde{s}'$ to Q;
13   **end**
14   **end**
15   **end**
16 **end**

4.2.2 Features and Decoding

We define our model in the log-linear framework (Och and Ney, 2002) over a derivation $D = r_1 r_2 \cdots r_N$, as in Equation (4.2):

$$p(D) \propto \prod_i \phi_i(D)^{\lambda_i} \tag{4.2}$$

where r_i are translation rules, ϕ_i are features defined on derivations, and λ_i are feature weights. In our experiments, we use the standard 8 features: two translation probabilities $p(G(s)|t)$ and $p(t|G(s))$, two lexical translation probabilities $p_{lex}(s|t)$ and $p_{lex}(t|s)$, a language model $p(t)$ over a translation t, a rule penalty, a word penalty, and a distortion function d for distance-based reordering.

The calculation of the distortion feature d in our model is different from the one used in conventional phrase-based models, because we need to take discontinuity into consideration. In our model, we use a distortion function defined in Galley and Manning (2010) to penalize discontinuous phrases that have relatively long gaps, as in Equation (4.3):

$$\begin{aligned} d(\tilde{s}_i, \tilde{s}_{i-1}) =& |\tilde{s}_i^{start} - \tilde{s}_{i-1}^{end} - 1| \\ &+ \sum_{n=2}^{N} |\overline{s}_{i,n}^{start} - \overline{s}_{i,n-1}^{end} - 1| \end{aligned} \tag{4.3}$$

where superscripts $start$ and end denote the start and end positions of a phrase, respectively. $\tilde{s}_i = [\overline{s}_{i,1}, \cdots, \overline{s}_{i,N}]$ is a phrase which has $N-1$ gaps and thus consists of N continuous phrases $\overline{s}_{i,n}$. Figure 4.5 shows an example of calculating distortion values for both continuous

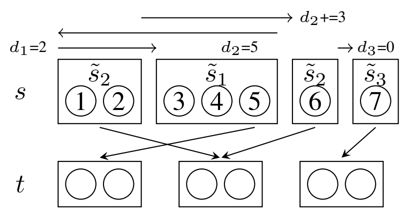


Figure 4.5: Distortion calculation for both continuous ($\tilde{s}_1$ and $\tilde{s}_3$) and discontinuous ($\tilde{s}_2$) phrases in a derivation.

and discontinuous phrases. According to Equation (4.3), the value of the distortion function d is a summation of two values. While the first one is the same as the distortion value in phrase-based models, the second one measures the length of gaps. In practice, instead of adding them together as a single distortion value, we treat the two values as two distinct features (Galley and Manning, 2010).

Our graph-based decoder is very similar to the phrase-based decoder, which generates hypotheses (partial translations) from left to right. In our decoder, however, each hypothesis is extended by translating an uncovered subgraph instead of a phrase. Positions covered by the subgraph are then marked as translated. The translation process ends when no untranslated words remain. Figure 4.6 shows a derivation of translating an input graph in Chinese to an English string.

As in the phrase-based model, beam search is taken as an approximation to reduce the search space. Hypotheses which cover the same number of source words are grouped in a stack. Hypotheses can be pruned according to their partial translation cost and an estimated future cost.

4.3 Graph Segmentation Model

Each derivation in our SegGBMT model implies a sequence of subgraphs (i.e. a segmentation). By default, similar to the phrase-based model, our model treats each segmentation equally as shown in Equation (4.1). However, previous work has suggested that such segmentations provide useful information which can improve translation performance. For example, boundary information in a phrase segmentation can be used for reordering models (Xiong et al., 2006; Cherry, 2013).

In this chapter, we are interested in directly modeling the segmentation using information from graphs. By making the assumption that each subgraph only depends on previous subgraphs, we define a generative process over a graph segmentation as in Equation (4.4):

$$p(G(\tilde{s}_{a_1}), \cdots, G(\tilde{s}_{a_I})) = \prod_{i=1}^{I} p(G(\tilde{s}_{a_i}) | G(\tilde{s}_{a_1}), \cdots, G(\tilde{s}_{a_{i-1}})) \tag{4.4}$$

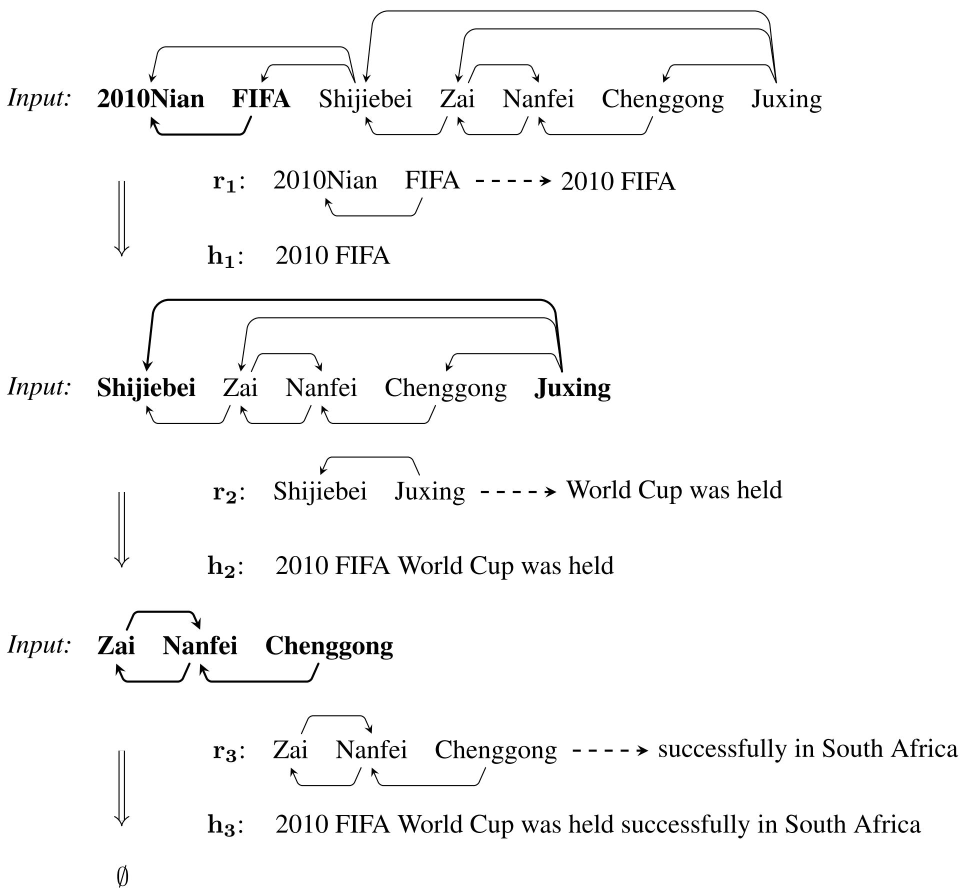
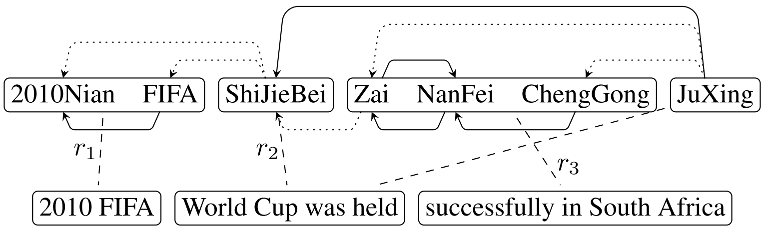


Figure 4.6: A derivation of translating an input graph. Each rule $\mathbf{r_i}$ matches an input subgraph (in bold) and generates a new hypothesis $\mathbf{h_i}$ by appending translations of the subgraph to the right.

Instead of training a stand-alone discriminative segmentation model to assign each subgraph a probability given previous subgraphs, we implement the model via sparse features, each of which is extracted at run-time with value 1 during decoding. Features with different names are directly added to the log-linear framework as new feature functions (whose values equal to their frequency) so that they can be tuned jointly with other default features (listed in Section 4.2.2) to directly maximize the translation quality.

Since a segmentation is obtained by breaking up the connectivity of an input graph, it is intuitive to use edges to model the segmentation. According to Equation (4.4), for a current subgraph $G_{\tilde{s}_{a_i}}$, we only consider those edges which are either inside $G_{\tilde{s}_{a_i}}$ or connect $G_{\tilde{s}_{a_i}}$



Sparse Features for r_2:

w=ShiJieBei@w=JuXing@p=C@d=in
w=ShiJieBei@c=1@p=C@d=in
w=ShiJieBei@w=2010Nian@p=P@d=out
w=ShiJieBei@c=2@p=C@d=out
w=ShiJieBei@w=FIFA@p=P@d=out
w=ShiJieBei@c=3@p=C@d=out
c=4@w=JuXing@p=C@d=in
c=4@c=1@p=C@d=in
c=4@w=2010Nian@p=P@d=out
c=4@c=2@p=C@d=out
c=4@w=FIFA@p=P@d=out
c=4@c=3@p=C@d=out
w=JuXing@w=ShiJieBei@p=C@d=out
w=JuXing@c=4@p=C@d=out
c=1@w=ShiJieBei@p=C@d=out
c=1@c=4@p=C@d=out

Figure 4.7: An illustration of extracting sparse features for each node in a subgraph during decoding. The class of a word represents a cluster ID and for simplicity is randomly assigned in this example: $\{$JuXing$\rightarrow$1, 2010Nian$\rightarrow$2, FIFA$\rightarrow$4, ShiJieBei$\rightarrow$4$\}$.

with a previous subgraph. Based on these edges, we extract sparse features for each node in the subgraph. The set of sparse features is defined as follows:

$$\left\{\begin{matrix} n.w \\ n.c \end{matrix}\right\} \times \left\{\begin{matrix} n'.w \\ n'.c \end{matrix}\right\} \times \left\{\begin{matrix} C \\ P \\ H \end{matrix}\right\} \times \left\{\begin{matrix} in \\ out \end{matrix}\right\}$$

where $n.w$ and $n.c$ are the word and class of the current node n, and $n'.w$ and $n'.c$ are the word and class of a node n' connected to n. C, P, and H denote that the node n' is in the current subgraph $G_{\tilde{s}_{a_i}}$ or the adjacent previous subgraph $G_{\tilde{s}_{a_{i-1}}}$ or other previous subgraphs $G_{\tilde{s}_{a_1}}, \cdots, G_{\tilde{s}_{a_{i-2}}}$, respectively. Note that we treat the adjacent previous subgraph differently from others since information from the immediately previous unit is quite useful (Xiong et al., 2006; Cherry, 2013). *in* and *out* denote that the edge is an incoming edge or outgoing edge for the current node n. Figure 4.7 shows an example of extracting sparse features for a subgraph.

Inspired by the success in using sparse features in SMT (Cherry, 2013), we lexicalize only on the top-100 most frequent words. In addition, we cluster source words into 50 classes by using *mkcls* (Och, 1999) and use cluster IDs to indicate classes of words which should provide useful generalization (Cherry, 2013) for our model.

4.4 Experiments

We implement our SegGBMT model in Moses. The graph segmentation model (GSM) is implemented as an optional component which is not included in SegGBMT by default. In addition to PBMT and HPBMT, we also compare SegGBMT with another two systems which are re-implemented in Moses for meaningful comparisons under the same settings:

- **Treelet** extends PBMT by treating treelets as the basic translation units (Menezes and Quirk, 2005; Quirk et al., 2005). We implement a Treelet system in Moses which produces translations from left to right and uses beam search for decoding.
- **DTU** extends PBMT by allowing discontinuous phrases without using linguistic structures (Galley and Manning, 2010). We implement DTU with source discontinuity in Moses.

We also conduct experiments to investigate translation performance of SegGBMT under different settings and compare SegGBMT with the D2S systems in Chapter 3. However, instead of consolidating all experimental results to a single table, we present a sequence of result tables so that we can clearly explain them one by one.

4.4.1 Basic Results

We first compare SegGBMT with PBMT, Treelet, and DTU. Table 4.2 shows evaluation results of the four systems. We found that SegGBMT is better than PBMT as measured by all three metrics across all test sets. Specifically, the improvements are +1.1/+0.8 (3.1%/3.9%, relative) BLEU, +0.2/+0.5 (0.5%/1.6%, relative) METEOR, and -0.3/-0.8 (0.5%/1.2%,

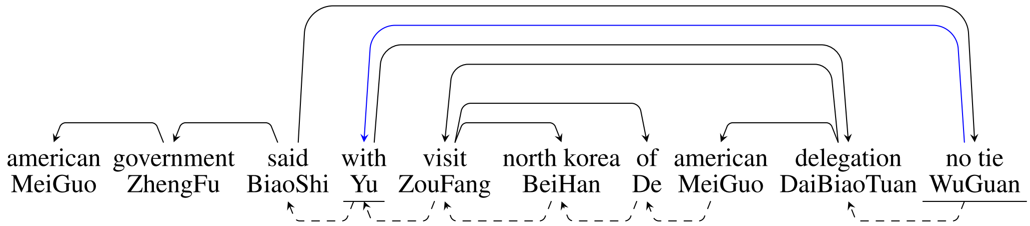
Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	PBMT	33.2	31.8^{+}	19.5	21.9
	Treelet	33.8^{*}	31.4	19.6	22.2^{+}
	DTU	34.7^{*+}	32.6^{*+}	19.7^{*}	22.4^{*}
	SegGBMT	34.7^{*+}	32.4^{*+}	$20.1^{*+\ddagger}$	$22.9^{*+\ddagger}$
METEOR ↑	PBMT	32.1	32.4+	28.0^{+}	29.2^{+}
	Treelet	32.0	32.0	27.8	29.0
	DTU	32.0	32.2^{+}	28.1^{*+}	29.5^{*+}
	SegGBMT	$32.4^{*+\ddagger}$	$32.4^{+\ddagger}$	$28.4^{*+\ddagger}$	$29.7^{*+\ddagger}$
TER ↓	PBMT	60.7	61.6^{+}	63.7	60.2
	Treelet	60.5	62.3	62.2^{*}	59.0^{*}
	DTU	59.3^{*+}	60.5^{*+}	63.2^{*}	59.3^{*}
	SegGBMT	60.1^{*+}	61.6^{+}	63.1^{*}	59.3^{*}

Table 4.2: Evaluation results of SegGBMT compared with PBMT, Treelet, and DTU. $*$ means a system is significantly better than PBMT at $p \leq 0.01$. $+$ means a system is significantly better than Treelet at $p \leq 0.01$. ‡ means SegGBMT is significantly better than DTU at $p \leq 0.01$.

relative) TER on average on ZH–EN and DE–EN, respectively. This improvement is reasonable as our system allows discontinuous phrases which can reduce data sparsity and handle long-distance relations (Galley and Manning, 2010). Figure 4.8 shows examples of translations where SegGBMT successfully translated a Chinese collocation *Yu... WuGuan* into *has nothing to do with*. By contrast, PBMT failed to catch the generalization since it only considers continuous phrases.

Compared to PBMT, the Treelet system does not show consistent improvements. Our system achieves significantly better BLEU (+1/+0.6) and METEOR (+0.4/+0.7) scores than Treelet on both ZH–EN and DE–EN, and a better TER score (-0.6) on ZH–EN. This suggests that continuous phrases are essential for system robustness since it helps to improve phrase coverage (Hanneman and Lavie, 2009). Figure 4.9 shows examples of translations where Treelet translated a discontinuous phrase *Dui ... Zuofa* only to one word *on* and therefore an important target word *practice* was dropped. By contrast, bigram relations allowed our system SegGBMT to translate a more proper phrase *De Zuofa* to *practice of*.

Compared with DTU, our system achieves comparable results. However, since discontinuous phrases produced by using syntactic information are fewer in number but more reliable

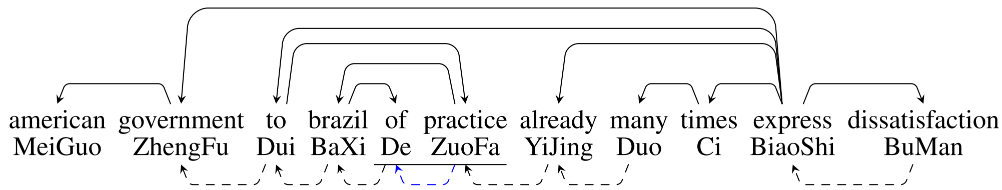


Ref: The american government said that it has nothing to do with the american delegation to visit north korea

PBMT: The government has said that the united states and north korea delegation has visited the united states

SegGBMT: The united states has indicated that it has nothing to do with the us delegation visited north korea

Figure 4.8: Examples of translations from SegGBMT and PBMT. SegGBMT successfully translated a Chinese collocation (underlined) into a target phrase. PBMT failed to capture this generalization because it only uses continuous phrases.



Ref: The us government has expressed their resentment against practice of brazil on many occasions

Treelet: the us government on many occasions brazil expressed dissatisfaction

SegGBMT: The us government has repeatedly expressed dissatisfaction with the practice of brazil

Figure 4.9: Examples of translations from SegGBMT and Treelet. By using bigram links, SegGBMT successfully translated the underlined continuous phrase which is not connected in the dependency tree.

System	# Rules	
	ZH–EN	DE–EN
DTU	224M+	352M+
SegGBMT	99M+	153M+

Table 4.3: The number of rules learned by DTU and SegGBMT.

(Quirk et al., 2005), our SegGBMT system uses significantly fewer rules. Table 4.3 shows the number of rules used in both systems. We found that DTU learns a huge model which contains more than twice as many rules compared to SegGBMT.

4.4.2 Influence of Rule Types

As described in Section 4.1, our model extracts rules on different types of subgraphs: (i) subgraphs connected by bigram links; (ii) subgraphs connected by dependency links; (iii) subgraphs which contain both types of links but are not connected by a single link type. The sets of rules on these subgraphs are called **PhrRule**, **TreeRule**, and **SpecRule**, respectively. In this section, we conduct experiments to examine the impact of different sets of rules on translation performance of our SegGBMT model.

Table 4.4 shows the number of rules in different rule sets. We found that in SegGBMT, $\sim$70% of rules are in the set PhrRule for both language pairs, which means that the source sides of most rules are connected by bigram links. Around 42%–48% of rules are connected by dependency links (TreeRule). We also observed that $>$30% of rules are shared by PhrRule and TreeRule. In terms of SpecRule, we found it contains about 15%–17% of all rules.

Table 4.5 shows BLEU scores of SegGBMT on different sets of rules. We found that while SegGBMT based on TreeRule (denoted as $\text{SegGBMT}_{\text{TreeRule}}$) achieves significantly better results (+0.3 BLEU) than $\text{SegGBMT}_{\text{PhrRule}}$ on DE–EN, it is worse on ZH–EN (-0.5 BLEU). This suggests that TreeRule is more useful on German sentences. This is probably due to the shorter MDD of German sentences than Chinese sentences (Eppler, 2013), which results in more rules extracted (42.7% vs 47.8%) and a higher phrase coverage on DE–EN

Rule Set	# Rules	
	ZH–EN	DE–EN
PhrRule	70M+	107M+
TreeRule	42M+	73M+
PhrRule+TreeRule	82M+	129M+
SpecRule	16M+	23M+
All	99M+	153M+

Table 4.4: The number of rules in different types in SegGBMT.

Rule Set	ZH–EN		DE–EN	
	MT04	MT05	WMT12	WMT13
PhrRule	34.4	32.3	19.6	22.0
TreeRule	33.8	32.0	19.8^{+}	22.4^{+}
+PhrRule	34.6^{*}	32.2	20.1^{+*}	22.9^{+*}
+SpecRule	34.7	32.4	20.1^{+}	22.9^{+}

Table 4.5: BLEU scores of SegGBMT using different sets of rules. $+$ means a system is significantly better than the system only based on PhrRule at $p \leq 0.01$. $*$ means a certain type of rules significantly improves a system at $p \leq 0.01$.

(Section 2.4). We also found that PhrRule improves $\text{SegGBMT}_{\text{TreeRule}}$ (+0.5 BLEU on ZH–EN and +0.4 BLEU on DE–EN, respectively), while SpecRule has no significant impact on translation performance. This is probably because SpecRule only introduces a small set of rules (15%–17%). By comparing these results with those in Table 4.2, we observed that $\text{SegGBMT}_{\text{PhrRule}}$ is better than PBMT as well (+0.9 BLEU on ZH–EN and +0.1 BLEU on DE–EN). This may suggest that phrase pairs extracted on continuous phrases have an ability to translate discontinuous phrases.

4.4.3 Statistics on Phrase Length

One of the arguments for discontinuous phrases is that they allow the decoder to use larger translation units which tend to produce better translations (Galley and Manning, 2010). However, this argument was only verified on ZH–EN. Therefore, we are interested in checking whether we have the same observation in our experiments on both language pairs.

We count the used translation rules in MT02 and WMT11 based on different target lengths. The results are shown in Figure 4.10. We found that both DTU and SegGBMT indeed tend to use larger translation units than both PBMT and Treelet on both language pairs. In addition, we found that Treelet uses more short phrases than other systems. This may suggest that Treelet has a lower phrase coverage as it discards continuous phrases which are not connected in trees.

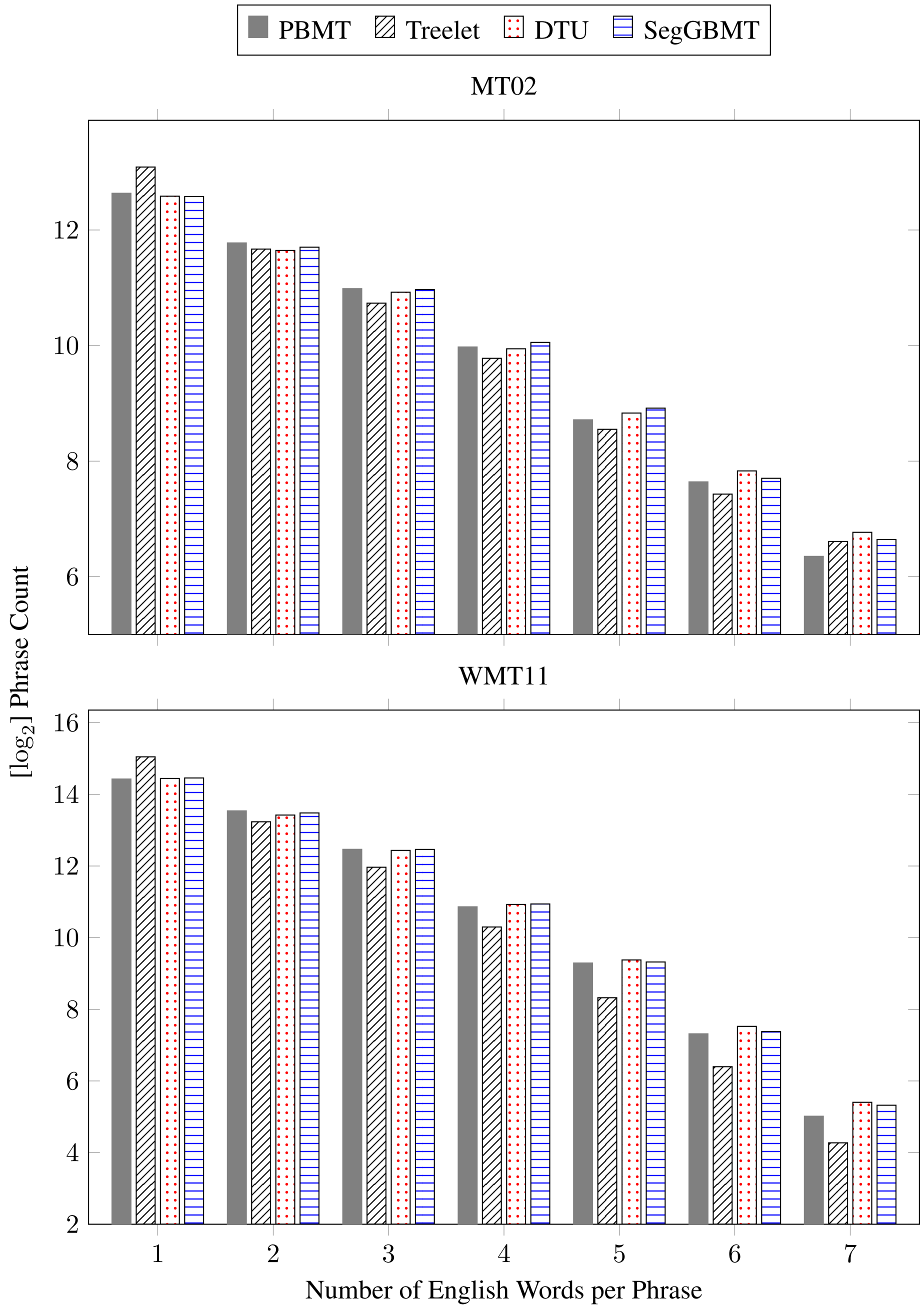
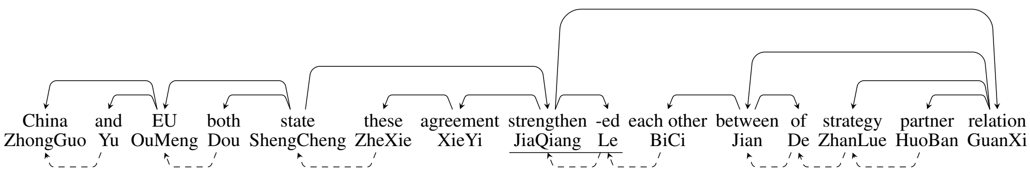


Figure 4.10: Phrase length histogram for MT02 and WMT11. x-axis represents phrase length, while y-axis denotes the number of phrases used. $\log_2$ is used to map values into a smaller range.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	SegGBMT	34.7	32.4	20.1	22.9
	+GSM	35.1*	32.6	20.4*	23.2*
METEOR ↑	SegGBMT	32.4	32.4	28.4	29.7
	+GSM	32.6*	32.5	28.6*	29.9*
TER ↓	SegGBMT	60.1	61.6*	63.1	59.3
	+GSM	60.2	62.1	62.7*	59.1*

Table 4.6: Evaluation results of SegGBMT using the graph segmentation model (GSM). $*$ means a system is significantly better than the other one at $p \leq 0.01$.



Ref: Both China and the EU claimed that these agreements have strengthened their strategic partnership

SegGBMT: China and the EU have claimed that these agreements to strengthen their mutual strategic partnership

SegGBMT+GSM: China and the EU have claimed that these agreements have strengthened their strategic partnership

Figure 4.11: Examples of translations from SegGBMT with the graph segmentation model (+GSM) which helped to select a better subgraph (underlined) to translate.

4.4.4 Influence of Graph Segmentation Model

As shown in Table 4.6, after integrating the GSM to help subgraph selection, SegGBMT is consistently improved across all four test sets (for instance, +0.3/+0.3 BLEU on average on ZH–EN and DE–EN, respectively). Figure 4.11 shows examples of translations of SegGBMT after integrating the segmentation model. We found that SegGBMT and SegGBMT+GSM generated translations which have the same prefix words. However, the segmentation model helped to select a subgraph covering *JiaQiang Le* to translate, which was subsequently translated into a better target phrase *have strengthened*.

However, our segmentation model is more helpful on DE–EN than on ZH–EN. We found that the number of features learned on ZH–EN (25K+) is much less than that on DE–EN (49K+). This resulted in a lower feature coverage during decoding (98% on DE–EN vs 95% on ZH–EN). The lower number of features in ZH–EN could be caused by the fact that the development set MT02 has many fewer sentences than WMT11 (878 vs 3,003). Accordingly, we suggest using a larger development set during tuning to achieve better translation performance when the segmentation model is integrated.

4.4.5 Integrating Lexical Reordering Model

One of the significant weaknesses of PBMT is phrase reordering. Although by default a distance-based reordering model is adopted, lexicalized reordering (LR) models (Koehn et al., 2005; Galley and Manning, 2008) are normally used to build a stronger system. However, in our preliminary experiments, PBMT+LR still performed worse than HPBMT which directly encodes reordering information in translation rules.

The weakness exists in SegGBMT as well. Therefore, we are interested in translation performance of our SegGBMT model when LR models are integrated. In experiments, we use a word-based LR model (Koehn et al., 2005), as in Equation (4.5):

$$p(o \mid G(\tilde{s}), \bar{t}) \approx p(o \mid \tilde{s}, \bar{t}) \tag{4.5}$$

where $o \in \{Monotonic, Swap, Discontinuous\}$ is an orientation. Figure 4.12 illustrates the differences between the three types of orientations.

The LR model introduces 6 features into SegGBMT. Table 4.7 shows evaluation results. We found that the LR model significantly improves SegGBMT. Specifically, the improvements on ZH–EN and DE–EN are +1.4/+0.5 (4.0%/2.3%, relative) BLEU, +0.4/+0.4

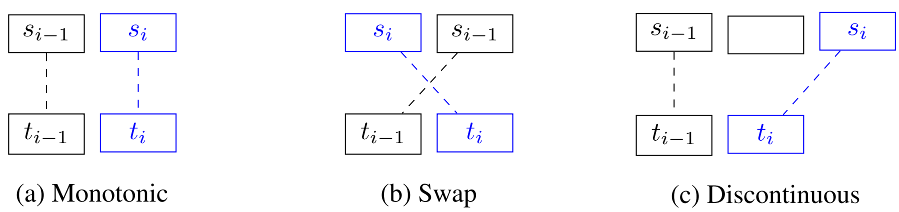


Figure 4.12: Illustration of three orientations for the phrase pair $\langle s_i, t_i \rangle$: (a) Monotonic; (b) Swap; (c) Discontinuous. Note that when an orientation of a phrase pair is neither Monotonic nor Swap, it is Discontinuous.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU $\uparrow$	HPBMT	36.5	34.3	20.5	23.0
	SegGBMT	34.7	32.4	20.1	22.9
	+LR	35.9*	33.9*	20.5*	23.5$*+$
METEOR $\uparrow$	HPBMT	33.0	33.2	28.5	29.8
	SegGBMT	32.4	32.4	28.4	29.7
	+LR	32.7*	32.9*	28.6*	29.9^{*+}
TER $\downarrow$	HPBMT	59.2	60.4	62.8	59.3
	SegGBMT	60.1	61.6	63.1	59.3
	+LR	59.2 *	60.5*	62.6^{*+}	58.8^{*+}

Table 4.7: Evaluation results of using a word-based lexical reordering (LR) model in SegGBMT. $*$ means SegGBMT+LR is significantly better than SegGBMT at $p \leq 0.01$. $+$ means a system is significantly better than HPBMT at $p \leq 0.01$.

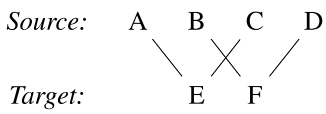


Figure 4.13: An illustration of a weakness in HPBMT compared with SegGBMT. When such crossing alignments exist, HPBMT is unable to independently generate translation units E and F.

(1.2%/1.2%, relative) METEOR, and -1.0/-0.5 (1.6%/0.8%, relative) TER. We also found that the improved SegGBMT is comparable with HPBMT on DE–EN. This is mainly because long-distance reordering is performed less often on DE–EN than on ZH–EN. Accordingly, stronger reordering capability is required on ZH–EN.

One of the advantages of SegGBMT over HPBMT is that it is much simpler than it (and other conventional syntax-based models). In addition, in theory the generalization capability of SegGBMT is less confined by crossing alignments (Galley and Manning, 2010), as shown in Figure 4.13. This is because HPBMT, as well as other conventional tree-based models, learns translation rules based on recursive grammars. These rules only cover continuous spans of an input sentence. Thus, when crossing alignments exist between translation units, it can only treat these units as a whole.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	SegGBMT	34.7	32.4	20.1	22.9
	+ET	34.7	32.7*	20.1	22.9
METEOR ↑	SegGBMT	32.4*	32.4*	28.4	29.7
	+ET	32.2	32.3	28.4	29.7
TER ↓	SegGBMT	60.1	61.6	63.1	59.3*
	+ET	59.0*	60.3*	63.2	59.4

Table 4.8: Evaluation results of SegGBMT with edge types (+ET). $*$ means a system is significantly better than another one at $p \leq 0.01$.

System	# Rules	
	ZH–EN	DE–EN
SegGBMT	99.2M+	153.4M+
+ET	99.7M+	153.8M+

Table 4.9: The number of rules in SegGBMT when edge types are included (+ET).

4.4.6 Influence of Edge Type

Our graphs consist of two types of edges: bigram edges and dependency edges. In our SegGBMT model, the two kinds of edges are used without distinction. However, it would be interesting to see how edge types impact on translation performance. Table 4.8 shows evaluation results of SegGBMT when edge types are taken into consideration (+ET) by adding them to the source side of rules. Results show that edge types do not consistently improve the SegGBMT system on the two language pairs. This is probably because edge types do not introduce many more rules into our system, as shown in Table 4.9.

4.4.7 Influence of Distortion Limit

A distortion limit is a parameter which disallows long-distance phrase reordering. It is adopted by PBMT to not only speed up the decoder but also as if often improve translation performance. Similarly, we used such a limit in our system as well. However, because our system translates subgraphs instead of phrases, which may cover discontinuous phrases, we are interested in how our system performs when the distortion limit changes.

Figure 4.14 shows BLEU scores of PBMT and SegGBMT when the distortion limit is set

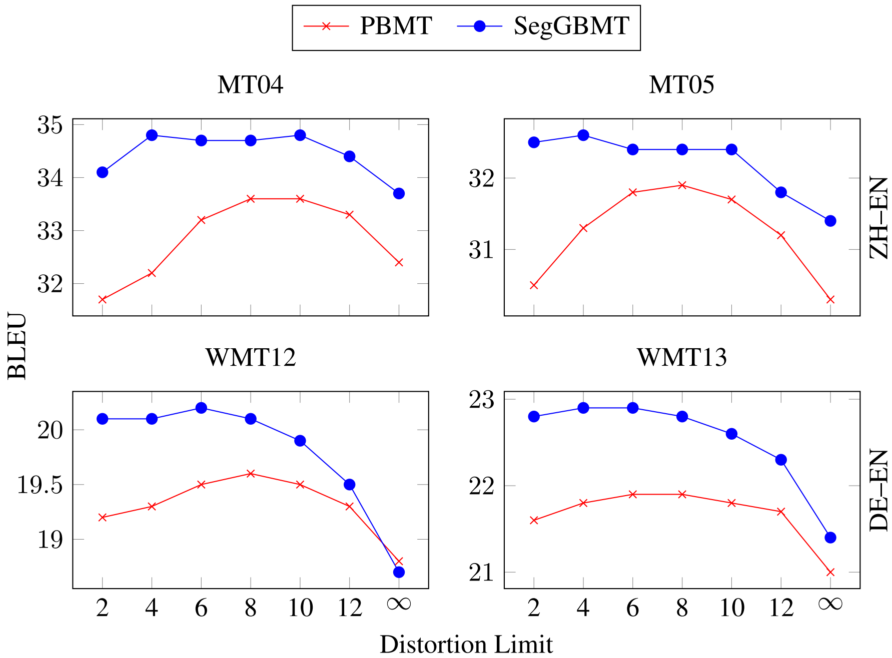
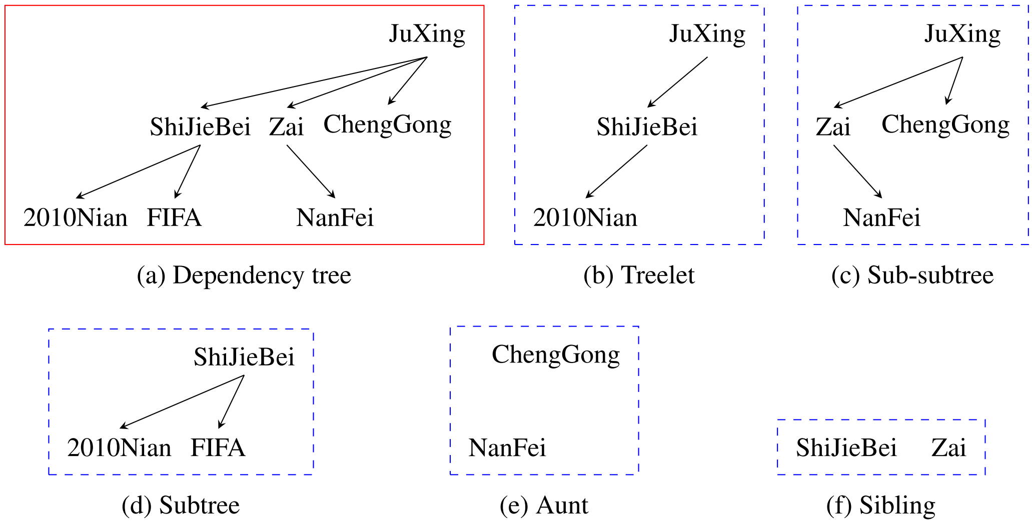


Figure 4.14: BLEU scores of PBMT and SegGBMT on all four test sets when distortion limit is set to different values.

to different values. We found that PBMT achieves its best performance when the distortion limit is between 6 and 10. When the limit is disabled (set to ∞) or is too small (≤ 2), PBMT achieves the worst BLEU scores. Compared with PBMT, SegGBMT performs better on different distortion values and is less sensitive to the parameter especially when its value is small (≤ 8). We presume the reason for this is that, even though the distortion limit is set to a small value, subgraphs can cover long-distance discontinuous phrases as long as they are connected in graphs.

4.4.8 Variance of Dependency Configurations

In this section, we compare SegGBMT with translation models in Chapter 3. We first compare their translation units which are covered by different dependency configurations. Given the dependency tree in Figure 4.15a, D2S in Chapter 3 can only cover *subtrees* as in Figure 4.15d. A subtree rooted at a node n contains all descendants of n. D2S+Decomp improves D2S by decomposing subtrees so that the system can cover *sub-subtrees*. A sub-



(a) Dependency tree (b) Treelet (c) Sub-subtree (d) Subtree (e) Aunt (f) Sibling

Figure 4.15: Given the dependency tree in (a), SegGBMT can cover dependency configurations (b)–(f).

subtree rooted at a node n contains only parts of subtrees of n. For example, Figure 4.15c shows a sub-subtree which excludes a subtree rooted at *ShiJieBei*. By contrast, in addition to subtrees and sub-subtrees, SegGBMT can also cover several other dependency configurations. Figure 4.15b is a *treelet* which is connected in the tree and covers a discontinuous phrase. The *aunt* configuration in Figure 4.15e and *sibling* configuration in Figure 4.15f are not connected in the tree. However, because they are continuous phrases, SegGBMT can cover them as well.

Table 4.10 shows evaluation results of SegGBMT and D2S+Decomp. We found that, compared with D2S+Decomp, SegGBMT is worse on the two language pairs. This is not surprising as SegGBMT is weak at phrase reordering (Section 4.4.5). When we integrate an LR model into SegGBMT, the resulting system SegGBMT+LR achieves better translation performance on DE–EN (+0.5 BLEU, +0.2 METEOR, -1.0 TER). However, on ZH–EN, SegGBMT systems are still worse than D2S systems. We presume the inconsistent system performance on the two language pairs is caused by more frequent long-distance reordering on ZH–EN than on DE–EN and SegGBMT is much weaker than D2S systems at phrase reordering as described in Section 4.4.5.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	D2S+Decomp	36.6	34.9	20.4	22.7
	+Phrase	37.7	35.5	20.8	23.4
	SegGBMT	34.7	32.4	20.1	22.9
	+LR	35.9	33.9	20.5	23.5^*
METEOR ↑	D2S+Decomp	32.1	32.1	28.4	29.7
	+Phrase	32.8	32.8	28.7	30.0
	SegGBMT	32.4^*	32.4^*	28.4	29.7
	+LR	32.7^*	32.9^*	28.6^*	29.9^*
TER ↓	D2S+Decomp	57.5	58.7	63.4	60.0
	+Phrase	56.9	58.0	62.7	59.1
	SegGBMT	60.1	61.6	63.1^*	59.3^*
	+LR	59.2	60.5	62.6^*	58.8^{*+}

Table 4.10: Evaluation results of SegGBMT compared with systems in Chapter 3. $*$ means SegGBMT (+LR) is significantly better than D2S+Decomp. $+$ means SegGBMT (+LR) is significantly better than D2S+Decomp+Phrase. Note that +Phrase is an extra resource added to D2S as described in Section 3.4.3.

4.5 Summary

In this chapter, we built graphs which combine bigram relations and dependency relations and presented a non-recursive graph-based translation model. The model translates a graph by segmenting it into subgraphs. In addition, we proposed a segmentation model which helps to select subgraphs. Our model can be regarded as a generalization of both phrase-based models and treelet-based models. This means that these models are special cases of our model: (i) by removing dependency edges, our model degrades to the phrase-based model; (ii) by removing bigram links, our model degrades to the treelet-based model. Experiments on ZH–EN and DE–EN show our model to be significantly better than both phrase-based and treelet-based models as well as another more sophisticated model DTU.

However, we also found that our model is weak at phrase reordering. In the next chapter, we will present graph-based models based on graph grammars. The model will learn recursive translation rules which directly encode reordering information.

Every speaker of a language has mastered and internalized a generative grammar that expresses his knowledge of his language.
— Noam Chomsky

Chapter 5

Recursive Graph-to-String Translation

In the previous chapter, we presented a non-recursive graph-based translation model which segments a graph into subgraphs and generates a complete translation by combining translations of these subgraphs. Although the model naturally covers both continuous and discontinuous phrases, phrase-reordering information is unavailable in rules.

In this chapter, we propose new models which use synchronous graph grammars to parse input graphs and simultaneously generate target strings. Translation rules in these models contain non-terminals which are used to specify how target phrases are reordered. In addition, we construct input graphs which implicitly or explicitly combine dependency relations and sibling relations, so that our models alleviate the weakness of tree-based models at handling non-syntactic phrases.

In the rest of this chapter, we first introduce two types of context-free graph grammars and present their formal definitions (Section 5.1). Then, we describe our graph-based models defined on the two graph grammars (Section 5.2 and 5.3). We discuss our experimental results in Section 5.4 followed by a short summary of this chapter (Section 5.5).

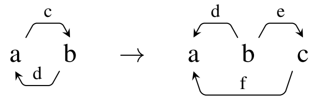
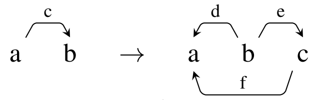
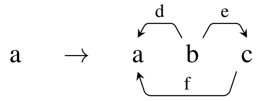
Grammar	Production Rules
graph replacement grammar	
edge replacement grammar	
node replacement grammar	

Table 5.1: Trivial examples of different types of production rules in graph grammars. For simplicity, indications of how to integrate graphs on the right-hand side into other graphs are ignored.

5.1 Graph Grammars

Similar to tree grammars which are used to generate trees, graph grammars are rewriting formalisms for generating graphs. In this chapter, all graphs and subgraphs are connected and directed (ordered). The production rule of the grammars is of the form $\langle\gamma, \alpha, \sim\rangle$, where γ is the left-hand side (possibly a graph, an edge, or a node) of the production, α is a graph, and $\sim$ indicates how to integrate the graph into another one (Kukluk et al., 2008).

In context-free grammars, the production rules can be applied without any restrictions on contexts which usually are vertices adjacent to γ or edges connected to it. When γ is a single edge, a single node, or a graph, we respectively call the grammar an edge, node, or graph replacement grammar (Kukluk et al., 2008). Table 5.1 shows trivial examples of production rules of the three grammars. In this chapter, we make use of Edge Replacement Grammar (ERG) and Node Replacement Grammar (NRG) to build translation models.

5.1.1 Edge Replacement Grammar

In this chapter, an ERG is defined as a context-free rewriting grammar to recognize and produce edge-labeled graphs which are connected, directed, and acyclic. Although in Definition 1.2, a graph is defined to be both edge-labeled and node-labeled, we do not consider node labels when introducing ERG as they can complicate our explanation (Chiang

(a) Edge-labeled graph (b) Node-labeled graph

Figure 5.1: Examples of an edge-labeled graph and a node-labeled graph

et al., 2013). In addition, we do not allow hyperedges to simply our explanation and translation model. We first redefine an edge-labeled graph as follows:

Definition 5.1. An *edge-labeled* and *ordered graph* is a tuple $H = \langle V, E, \phi \rangle$, where

- V is a finite set of nodes.
- $E \subseteq V^2$ is a finite set of edges.
- $\phi : E \rightarrow C$ is a function which assigns a label from C to each edge.

Figure 5.1a shows an example of an edge-labeled graph. In ERG, the elementary units are edge-labeled graph fragments, which are also the right-hand sides of production rules in the grammar. Its definition is as follows:

Definition 5.2. An *edge-labeled graph fragment* is a tuple $H = \langle V, E, \phi, X \rangle$, where $\langle V, E, \phi \rangle$ is an edge-labeled graph and X is one or two nodes in V. Following Drewes et al. (1997) and Chiang et al. (2013), we call these *external nodes*.

The external nodes indicate how to integrate a graph into another one during a derivation. ERG limits the number of external nodes to 2 at most to make sure hyperedges do not exist during a derivation. All edge-labeled graphs in this chapter are connected and acyclic. Therefore, there are one or more root nodes, which have no incoming edges, in an edge-labeled graph. Root nodes are always external nodes (Chiang et al., 2013). Based on edge-labeled graph fragments, we define the ERG as in Definition 5.3.

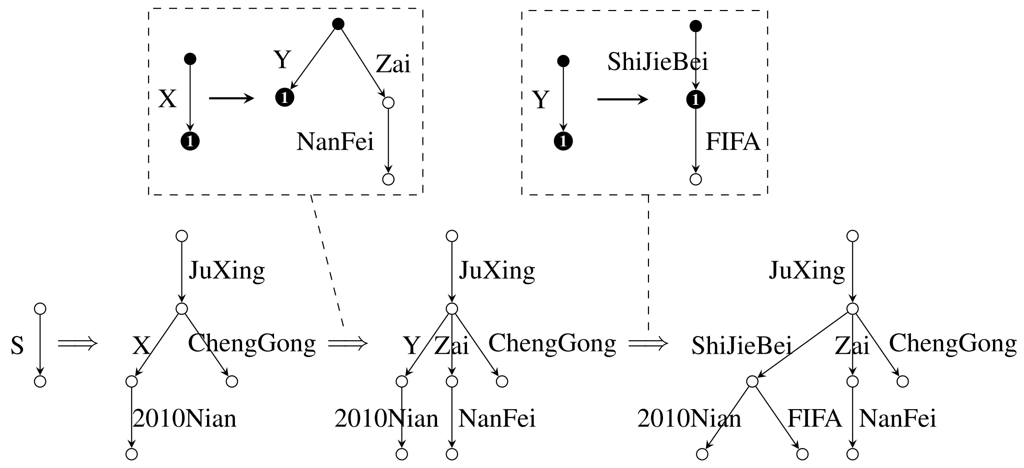


Figure 5.2: Illustration of a derivation in an ERG. Productions are in rectangles. X and Y are non-terminals. Dark circles are external nodes. Integers in external nodes indicate how to integrate the graph on the right-hand side when an edge is replaced.

Definition 5.3. An *edge replacement grammar* is a tuple $\langle N, T, P, S \rangle$, where

- N and T are disjoint finite sets of non-terminal symbols and terminal symbols, respectively.
- P is a finite set of productions of the form $(A \rightarrow R)$, where $A \in N$ and R is an edge-labeled graph fragment, where edge-labels are from $N \bigcup T$.
- $S \in N$ is the start symbol.

Figure 5.2 shows an example of a derivation in an ERG to produce a graph. Starting from the start symbol S, when a rule $(A \rightarrow R)$ is applied to an edge e, the edge is replaced by the graph fragment R. The ordering of nodes V_e in e and external nodes X_R in R implies the mapping from V_e to X_R (Drewes et al., 1997; Chiang et al., 2013).

To build a translation model, we need a Synchronous Edge Replacement Grammar (SERG), which is defined in Definition 5.4, to parse an input graph and simultaneously generate translations.

Definition 5.4. An *SERG* is a tuple $\langle N, T, T', P, S \rangle$, where

- N is a finite set of non-terminal symbols.

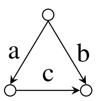
- T and T' are finite sets of terminal symbols.
- $S \in N$ is the start symbol.
- P is a finite set of productions of the form $(A \rightarrow \langle R, R', \sim \rangle)$, where $A \in N$, R is an edge-labeled *graph* fragment over $N \bigcup T$ and R' is an edge-labeled *graph* fragment over $N \bigcup T'$. $\sim$ is a one-to-one mapping between non-terminal symbols in R and R'.

Proposition 5.1. SERG has a stronger generative capacity over structure pairs than both SCFG (Definition 2.1) and STSG (Definition 2.2).

Proof. STSG has a stronger generative capacity over structures than SCFG (Chiang, 2012). For example, the following STSG generates a trivial example of a tree pair that no SCFG can generate, as SCFG must always have an equal number of non-terminal symbols on each side of the tree pair (Chiang, 2012).

$$\begin{array}{c} X \\ | \\ X \\ | \\ \epsilon \end{array} : \begin{array}{c} X \\ | \\ \epsilon \end{array}$$

Any STSG can easily be converted into an SERG by labeling edges in tree structures. The following SERG generates a trivial example of a graph pair, which no STSG can generate, as the left structure is a graph rather than a tree.

$X \rightarrow$  : $X \rightarrow$ c

$\square$

5.1.2 Node Replacement Grammar

Different from ERG, NRG in this chapter generates node-labeled graphs which are connected and directed as in Definition 5.5. By default, we do not consider edge labels in this section as they have no significant impact on our system in experiments (Table 5.6 in Section 5.4.5).

Definition 5.5. A *node-labeled* and *ordered graph* is a tuple $H = \langle V, E, \phi \rangle$, where

- V is a finite set of nodes.
- $E \subseteq V^2$ is a finite set of edges.
- $\phi : V \to C$ is a function which assigns a label from C to each node.

Figure 5.1b shows an example of a node-labeled graph. Similar to ERG, in NRG the elementary units are node-labeled graph fragments, which are also the right-hand sides of production rules in the grammar. Its definition is as follows:

Definition 5.6. A *node-labeled graph fragment* is a tuple $H = \langle V, E, \phi, X \rangle$, where $\langle V, E, \phi \rangle$ is a node-labeled graph and X is an embedding mechanism which is a set of connection instructions to indicate how to integrate a graph into another one.

Note that the embedding mechanism is important in the generation process (as in Figure 5.3), but since in graph-to-string translation models graph grammars are used to parse source graphs, the embedding mechanism will be omitted in this chapter (Kukluk, 2007).

Definition 5.7. A *node replacement grammar* is a tuple $\langle N, T, P, S \rangle$, where

- N and T are disjoint finite sets of non-terminal symbols and terminal symbols.
- P is a finite set of productions of the form $(A \to R)$. $A \in N$ and R is a node-labeled graph fragment where node labels are from $N \bigcup T$.
- $S \in N$ is the start symbol.

Figure 5.3 shows an example of a derivation in an NRG to produce a node-labeled graph. Starting from the start symbol S, when a rule $(A \to R)$ is applied to a node v, the node together with edges linked to it is removed and the graph fragment R is inserted. The embedding mechanism in R includes node pairs associated with edge directions (Rozenberg, 1997) to indicate how to add connections between neighbors of v and nodes in R.

Similar to SERG, a Synchronous Node Replacement Grammar (SNRG) which can be used in SMT is defined in Definition 5.8.

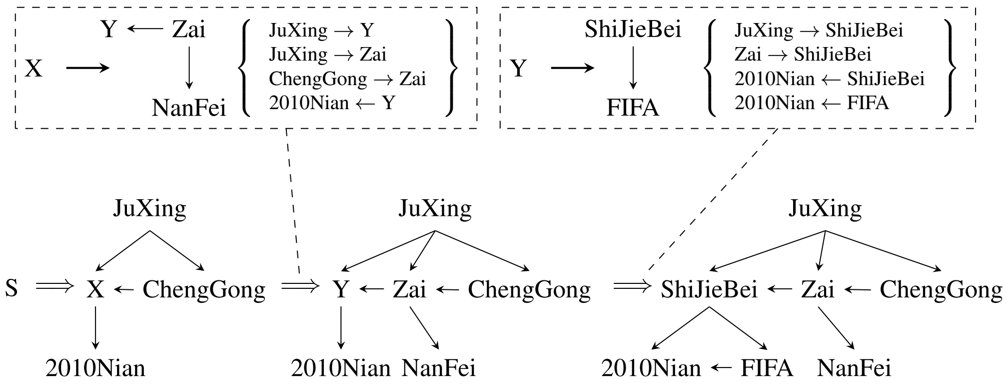


Figure 5.3: Illustration of a derivation in an NRG. Productions are in rectangles. Embedding mechanisms for integrating graphs are indicated by $\{\}$. For example, *JuXing*→*Y* means adding an edge from *JuXing* to *Y* if *JuXing* is one of its neighbors.

Definition 5.8. An *SNRG* is a tuple $\langle N, T, T', P, S \rangle$, where

- N is a finite set of non-terminal symbols.
- T and T' are finite sets of terminal symbols.
- $S \in N$ is the start symbol.
- P is a finite set of productions of the form $(A \to \langle R, R', \sim \rangle)$, where $A \in N$, R is a node-labeled *graph* fragment over $N \bigcup T$ and R' is a node-labeled *graph* fragment over $N \bigcup T'$. $\sim$ is a one-to-one mapping between non-terminal symbols in R and R'.

5.2 ERG-Based Translation

In this section, we present a translation model based on SERG which translates edge-labeled graphs into target strings. This means that translation rules in our model are graph–string pairs instead of graph–graph pairs, which are in the form of (5.1):

$$\langle N(\gamma) \to \gamma, X \to \alpha, \sim \rangle \tag{5.1}$$

where γ is an edge-labeled graph on the source side which is connected, $N(\gamma)$ is a source non-terminal which will be introduced in Section 5.2.2, X is the general non-terminal on

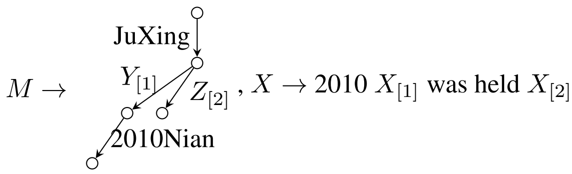


Figure 5.4: An example of a rule translating an edge-labeled graph into a target string. X is the general non-terminal on the target side. M, Y, and Z are source non-terminals. Indexes indicate mappings between source and target non-terminals.

the target side, α is a string over target terminals and non-terminals, and $\sim$ is a one-to-one mapping between source and target non-terminals. Figure 5.4 shows an example of a translation rule. Note that in the rule indications of external nodes are ignored as they are not required during decoding (Kukluk, 2007).

5.2.1 Dependency-Edge Graphs

Before introducing our ERG-based translation model, we first describe how to construct edge-labeled graphs. In this chapter, an edge-labeled graph is directly derived from a dependency tree by labeling edges with words, as shown in Figure 5.5. The graph is called a Dependency-Edge Graph (DEG). The DEG has an advantage over the dependency tree: siblings are connected in the DEG. Therefore, non-syntactic phrases, such as *2010Nian FIFA* in the dashed rectangle in Figure 5.5, are connected and thus can be covered by our model.

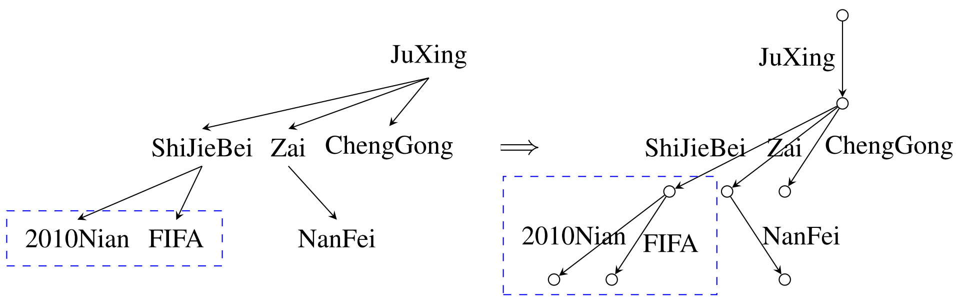


Figure 5.5: Illustration of converting a dependency tree into a dependency-edge graph where siblings, such as *2010Nian FIFA*, are connected.

(a) DEG over *C B D* (b) DEG over *B C D*

Figure 5.6: Two dependency-edge graphs (DEGs) with different word order.

5.2.2 Practical Restrictions

To efficiently and effectively use SERG in SMT, we adopt several restrictions.

Word-Order Restriction

In Definition 5.1 in Section 5.1.1, edge-labeled graphs are defined without considering word order. However, word order is an important piece of information for SMT. Therefore, we add a word-order restriction to DEGs. The word-order in DEGs is indicated by relative positions of nodes and edges. For example, Figure 5.6 shows two different DEGs which cover different sentences.

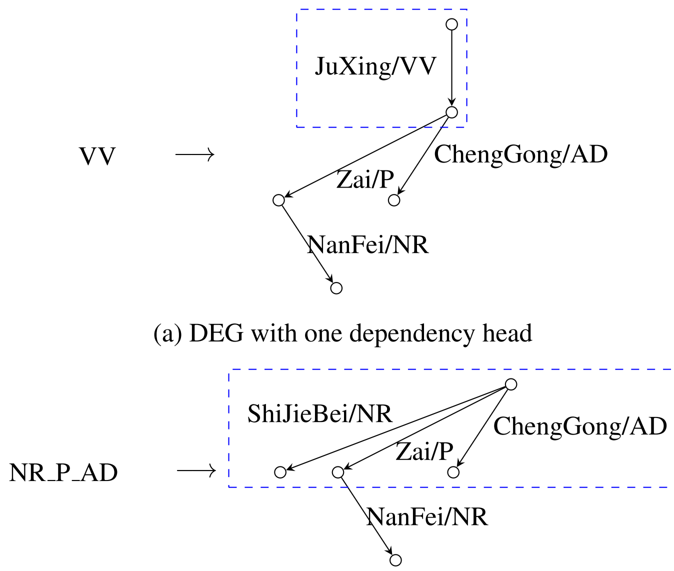
Continuity Restriction

The time complexity of decoding a DEG depends on the number of subgraphs in the DEG. Algorithm 5.1 shows an abstract decoding process. The decoder needs to access all subgraphs $\overline{g}$ of a graph g (Lines 1) and then applies rules to each subgraph to generate hypotheses (Lines 2–7). Assuming we have a graph with k nodes, the number of all possible subgraphs could be $O(2^k)$. Therefore, if we take all subgraphs into consideration, the time complexity of decoding would be exponential to the graph size.

It is easy to find that the large number of subgraphs is caused by the free combination of edges. This means that subgraphs in an input sentence may cover any discontinuous spans. Therefore, for efficiency, we add a restriction to our ERG-based model: translation rules only cover continuous spans of an input sentence. This reduces the complexity of our decoding algorithm from exponential time to cubic time as in tree-based models.

Algorithm 5.1: Sketch of a decoding process.

```
Data: input graph g
Result: translation t
forall subgraphs ḡ ∈ g do
    forall rules r do
        if r can be applied to ḡ then
            create new hypothesis h
            add h to chart
        end
    end
end
```



(a) DEG with one dependency head

(b) DEG with multiple dependency heads

Figure 5.7: Illustration of inducing non-terminal symbols (left side) for DEG fragments (right side). Each edge is labeled with a word associated with its POS tag. The head of a fragment is included in a dashed rectangle.

Non-terminal Restriction

Since our model translates input DEGs into target strings, we only use a general non-terminal symbol X as in HPBMT (Chiang, 2005; Chiang, 2007) on the target side. However, previous work suggested that HPBMT benefits from replacing the X with linguistic non-terminals (Zollmann and Venugopal, 2006; Almaghout et al., 2011; Almaghout et al., 2012; Li et al., 2012b). Therefore, on the source side of translation rules, we use non-terminal symbols based on POS tags, which can be easily obtained as a by-product of dependency parsing.

We define the *head* of a DEG g as a list of edges, the dependency head (head in the dependency tree) of each of which is not in g. Then, the non-terminal symbol for g is defined as the joining of POS tags of its head (Li et al., 2012b). Figure 5.7 shows examples of assigning non-terminal labels to DEGs. When a DEG has only one dependency head (Figure 5.7a), we simply use its POS tag as the non-terminal. When a DEG has multiple dependency heads (Figure 5.7b), we use a joint POS tag as the non-terminal.

5.2.3 Rule Extraction

As well as the restriction defined in Section 5.2.2 making the grammar much smaller, it also results in a similar way of extracting translation rules as in HPBMT. Inspired by HPBMT, we define the rule set over *initial pairs*. Given a word-aligned dependency graph–string pair $P = \langle g, t, a \rangle$, a pair $\langle \overline{g}, \overline{t} \rangle$ is an initial pair of P, iff:

1. $\overline{g}$ is a connected subgraph of g covering a continuous source phrase and has at most two external nodes. $\overline{t}$ is a continuous target phrase.

2. It is consistent with the word alignment a (Och and Ney, 2004).

The set of rules from P satisfies the following:

1. If $\langle \overline{g}, \overline{t} \rangle$ is an initial pair, then

$$\langle N(\overline{g}) \to \overline{g}, X \to \overline{t} \rangle$$

is a rule, where $N(\overline{g})$ returns a non-terminal symbol for $\overline{g}$ as defined in Section 5.2.2.

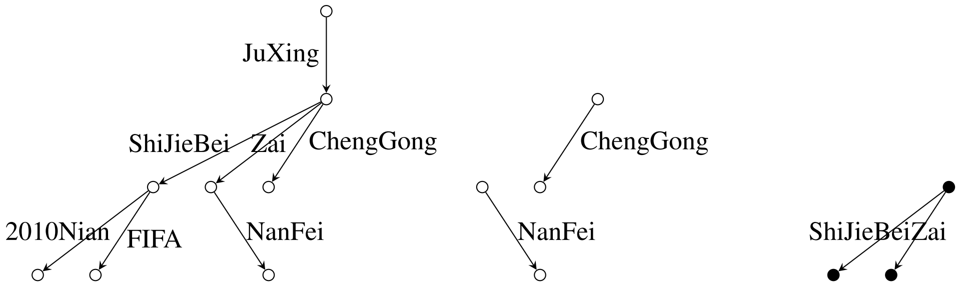
2. If $\langle N(R) \rightarrow R, X \rightarrow R' \rangle$ is a rule of P and $\langle \overline{g}, \overline{t} \rangle$ is an initial pair such that $\overline{g}$ is a subgraph of R and $R' = r_1\overline{t}r_2$, then

$$\langle N(R) \rightarrow R \backslash \overline{g}_{[k]}, X \rightarrow r_1 X_{[k]} r_2 \rangle$$

is a rule of P, where $\backslash$ means replacing $\overline{g}$ in R with an edge labeled with $N(\overline{g})$ and k is a unique index for a pair of non-terminal symbols.

We use an extraction algorithm similar to the one in HPBMT and use glue rules to monotonically combine graph fragments and translations when no matched rule can be found. However, the difference is that we need to check if a source span is covered by a valid graph fragment, in which case we keep the graph structure and induce a non-terminal for the fragment. Otherwise, we do not extract rules on this source span. In this sense, our model extracts fewer rules than HPBMT. For example, given the DEG in Figure 5.8a, HPBMT extracts rules whose source sides cover any continuous phrases (e.g. Figure 5.8b) as long as the phrase is consistently aligned to a target phrase. However, our model cannot make use of this phrase in Figure 5.8b because it is not connected in the DEG. In addition, we cannot extract rules on the fragment in Figure 5.8c, because it has more than two external nodes.

Algorithm 5.2 provides a sketch of our extraction algorithm. The algorithm traverses all valid source subgraphs $\overline{g}$ within a size limitation l_{max} where the number of external nodes $\overline{g}.ext$ is less than or equal to 2 (Line 2–4). Then, for each target phrase $\overline{t}$ which is consistently aligned to $\overline{g}$, we extract a lexicalized rule r (Lines 6–7). Then, we extract recursive rules (Line 9). Note that, to extract recursive rules, we need a set H to store the spans of all extracted rules so far (Line 8). These spans specify non-terminal positions (Lines 18). $\overline{g} \backslash \overline{g}_h$ means replacing the subgraph $\overline{g}_h$, which covers the source span in h, with an edge labeled with $N(\overline{g}_h)$, while $\overline{t} \backslash \overline{t}_h$ means replacing the phrase $\overline{t}_h$, which covers the target span in h, with the general non-terminal X. Finally, we recursively call the function $RecurRule(r)$ to generate all possible recursive rules (Line 21).



(a) DEG (b) unconnected fragment (c) fragments with three external nodes

Figure 5.8: Given the DEG in (a), our ERG-based translation model cannot cover the two phrases in (b) and (c), because (b) is not connected and (c) has more than two external nodes (in dark).

Algorithm 5.2: Algorithm for extracting translation rules from a graph–string pair in our ERG-based translation model.

Data: graph–string pair $\langle g, t, a\rangle$
Result: translation rules P

$H \leftarrow \{\}, P \leftarrow \{\}$
for $l \leftarrow 1$ to l_{max} **do**
    forall $\overline{g}$ in g: $|\overline{g}| \leq l$ **do**
        if $|\overline{g}.ext| \leq 2$ **then**
            forall $\overline{t}$ which is consistently aligned to $\overline{g}$ **do**
                $r \leftarrow \langle N(\overline{g}) \rightarrow \overline{g}, X \rightarrow \overline{t}\rangle$
                add r to P
                add $\langle Span(\overline{g}), Span(\overline{t})\rangle$ to H
                `RecurRule`(r)
            end
        end
    end
end
Function `RecurRule`($r = \langle N(\overline{g}) \rightarrow \overline{g}, X \rightarrow \overline{t}\rangle$)
    if Continue(r) **then**
        forall $h \in H$ **do**
            if h covers a terminal subspan of r **then**
                $\overline{g'} \leftarrow \overline{g} \backslash \overline{g}_h, \overline{t'} \leftarrow \overline{t} \backslash \overline{t}_h$
                $r' \leftarrow \langle N(\overline{g}) \rightarrow \overline{g'}, X \rightarrow \overline{t'}\rangle$
                add r' to P
                `RecurRule`(r')
            end
        end
    end
end

5.2.4 Features and Decoding

We define our model in the log-linear framework over a derivation d, as in Equation (5.2):

$$p(d) \propto \prod_i \phi_i(d)^{\lambda_i} \tag{5.2}$$

where ϕ_i are features defined on derivations and λ_i are feature weights. In our experiments, we use the standard 8 features: translation probabilities $p(g|t)$ and $p(t|g)$, lexical translation probabilities $p_{lex}(g|t)$ and $p_{lex}(t|g)$, a language model $p(t)$ over a translation t, rule penalty $exp(-1)$, word penalty $exp(|t|)$, and glue rule penalty $exp(-1)$.

Our decoder is based on the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970). It searches for the best derivation among all possible derivations. For each continuous span of an input graph, the decoder checks if it is covered by a valid graph fragment. Then, for each valid fragment, the decoder finds rules to translate it. The translation of a large span can be obtained by combining translations from its sub-spans using rules which contain non-terminals. Finally, glue rules are used to make sure that at least one translation is produced.

Figure 5.9 shows a derivation which parses a Chinese DEG and simultaneously generates an English string. When a rule is applied, a DEG fragment in the source graph is replaced by an edge, and a new hypothesis is generated. Non-terminals in the target string of the rule are replaced by previous hypotheses.

We found that in Figure 5.9 the second rule r_2 translates a non-syntactic phrase *Zai NanFei ChengGong*, which can be a problem for conventional tree-based models. In addition, the first rule r_1 translates a treelet and the third rule r_3 specifies how to reorder target phrases. All three aspects are uniformly represented in our model, which makes it more powerful than other methods, such as the treelet approach (Menezes and Quirk, 2005; Quirk et al., 2005; Xiong et al., 2007) and the Dep2Str model (Xie et al., 2011).

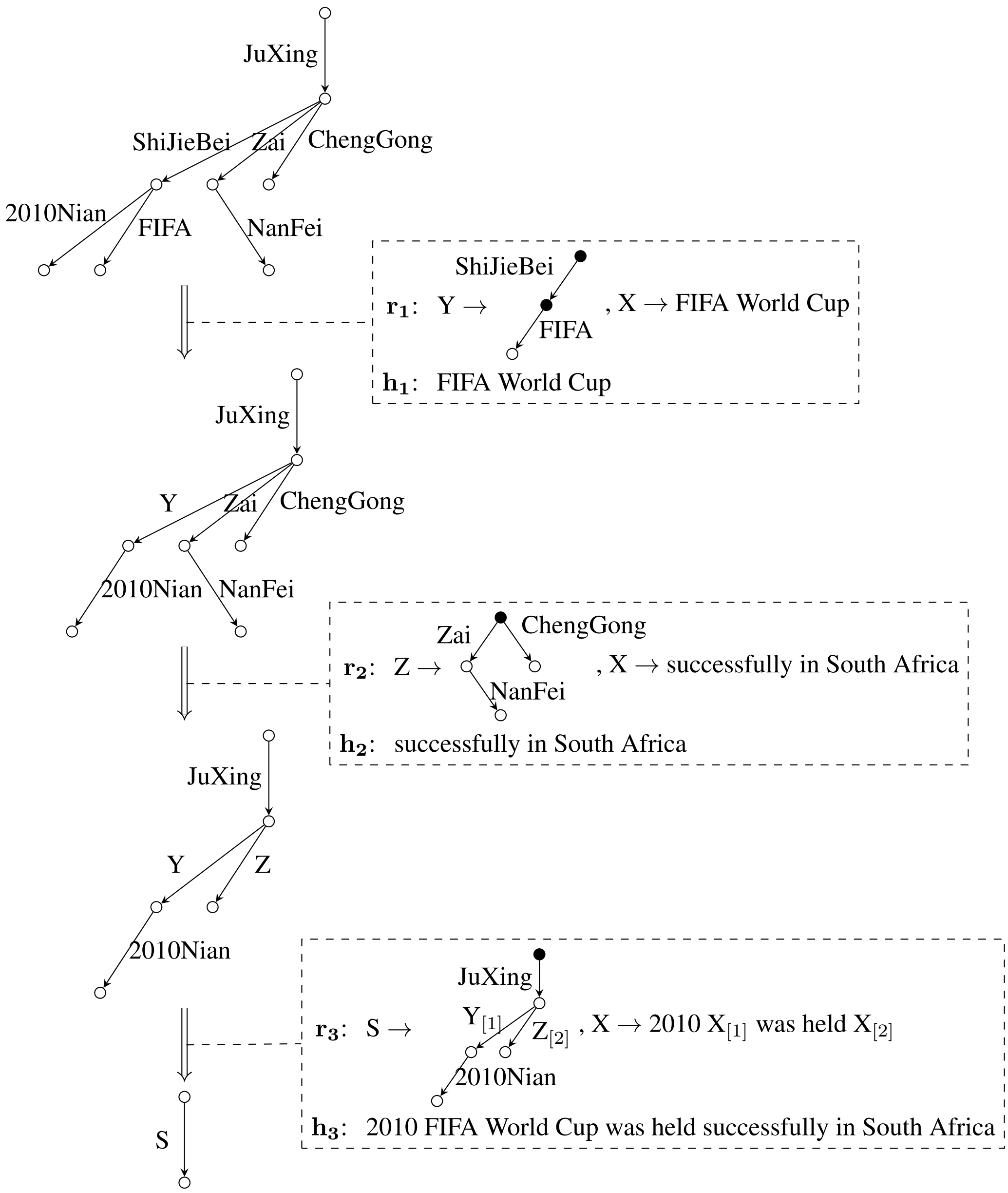


Figure 5.9: An example of a derivation in our ERG-based model to parse a DEG over a Chinese sentence *2010Nian FIFA ShiJieBei Zai NanFei ChengGong JuXing* and generate an English string. r_i are rules while h_i are hypotheses. External nodes in dark circles are implied during decoding. In addition to the start symbol S, non-terminal symbols for the source side are Y and Z, while the target side only has one non-terminal X. Indexes on non-terminals of rules indicate mappings.

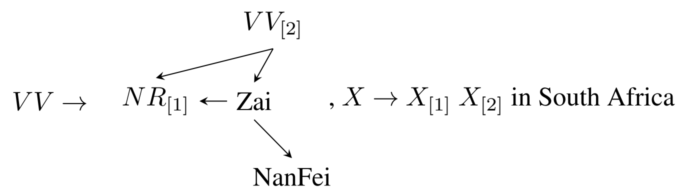


Figure 5.10: An example of a translation rule in our NRG-based model. VV and NR are source non-terminals (defined in Section 5.2.2) while X is a general target non-terminal. Indexes indicate mappings between source and target non-terminals.

5.3 NRG-Based Translation

Similar to the ERG-based translation model which translates edge-labeled graphs, we use an NRG-based model to translate node-labeled graphs. Rules in our NRG-based model are in the form of (5.3):

$$\langle N(\gamma) \rightarrow \gamma, X \rightarrow \alpha, \sim\rangle \tag{5.3}$$

where γ is a node-labeled source graph which is connected, $N(\gamma)$ is a source non-terminal which has been introduced in Section 5.2.2, X is the general non-terminal on the target side, α is a string over target terminals and non-terminals, and $\sim$ is a one-to-one mapping between source and target non-terminals. Figure 5.10 shows an example of a translation rule.

5.3.1 Dependency-Sibling Graphs

The node-labeled graph used in this chapter is called a Dependency-Sibling Graph (DSG) which is generated by adding sibling links to a dependency tree. Figure 5.11 shows an example of a DSG. Each node is labeled with an input word. By directly connecting siblings, our model can cover non-syntactic phrases, such as *ShiJieBei Zai* and *Zai NanFei ChengGong* et al.

Different from a DBG as defined in Chapter 4 which includes links between any two consecutive words, a DSG only adds links to two consecutive siblings which may be discontinuous in sequences. Although our NRG-based model can take DBGs as inputs, we do not use them in this chapter, because the continuity restriction in Section 5.2.2 makes our

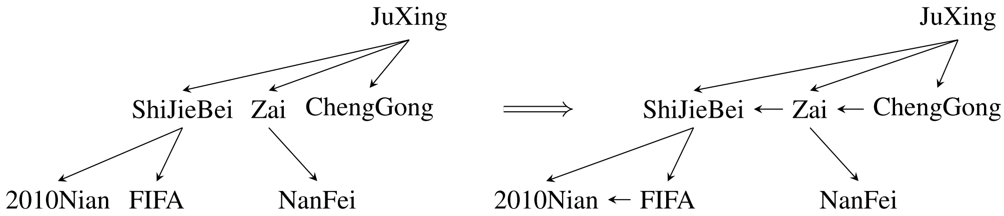


Figure 5.11: Converting a dependency tree to a dependency-sibling graph by directly adding edges between siblings.

model only cover continuous spans which are all connected in DBGs. This means that, if we use DBGs as inputs, checking the graph connectivity during training and decoding will be meaningless.

5.3.2 Training and Decoding

We still adopt the restrictions in Section 5.2.2 in our NRG-based model: a DSG keeps word-order, each translation rule covers a continuous span of an input sentence, and we use POS tags to define non-terminals. Therefore, a training process similar to the one in Section 5.2.3 can be used to extract NRG-based rules.

We define our NRG-based model in the log-linear framework and use the same features as in Section 5.2.4. The decoder is based on the CYK algorithm which bottom-up translates an input graph. Each rule is applied to a continuous source span which is covered by a connected subgraph. The translation process ends when all valid spans have been visited. Figure 5.12 shows a derivation to translate a Chinese DSG into an English string.

5.4 Experiments

We conduct experiments on ZH–EN and DE–EN language pairs. Our recursive graph-based systems are implemented in Moses and are called **SERG** and **SNRG**, respectively. In addition to comparing our systems with HPBMT which is better than PBMT, we also conduct experiments to investigate translation performance of our systems under different settings and compare them with the D2S systems in Chapter 3 and SegGBMT systems in

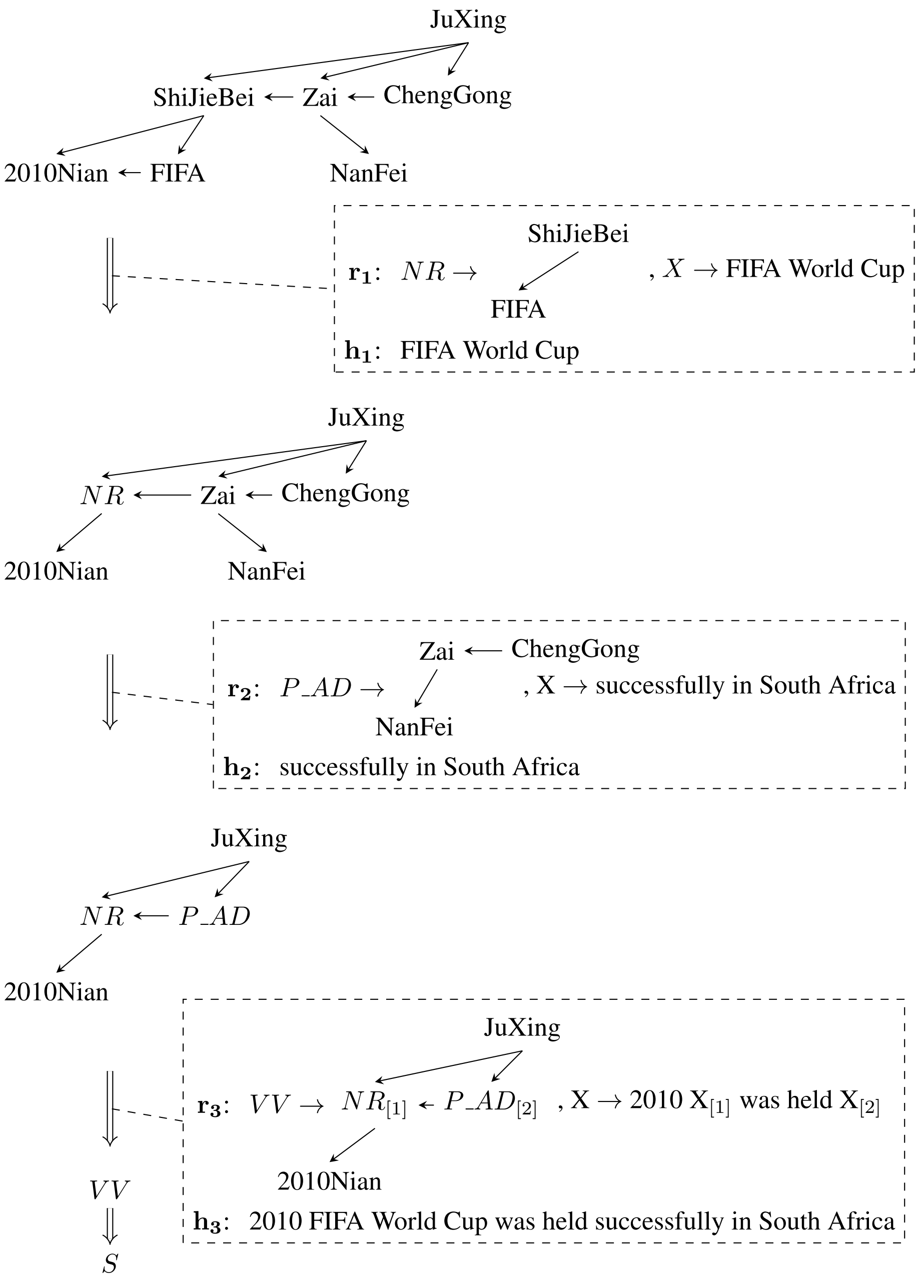


Figure 5.12: An example of a derivation in our NRG-based model which parses a DSG over a Chinese sentence *2010Nian FIFA ShiJieBei Zai NanFei ChengGong JuXing* and generates an English string. r_i are rules while h_i are hypotheses. In addition to the start symbol S, non-terminal symbols for the source side are NR, P_AD, and VV, while the target side only has one non-terminal X. Indexes on non-terminals of rules indicate mappings.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	HPBMT	36.5	34.3	20.5	23.0
	SERG	37.7*	35.8*	20.6	23.2*
	SNRG	37.7*	35.8*	20.7	23.4*
METEOR ↑	HPBMT	33.0	33.2	28.5	29.8
	SERG	32.8	33.0	28.7*	30.0*
	SNRG	32.6	32.7	28.7*	30.0*
TER ↓	HPBMT	59.2	60.4	62.8	59.3
	SERG	56.7*	57.9*	62.7	59.1*
	SNRG	56.2*	57.3*	62.5*	59.1*

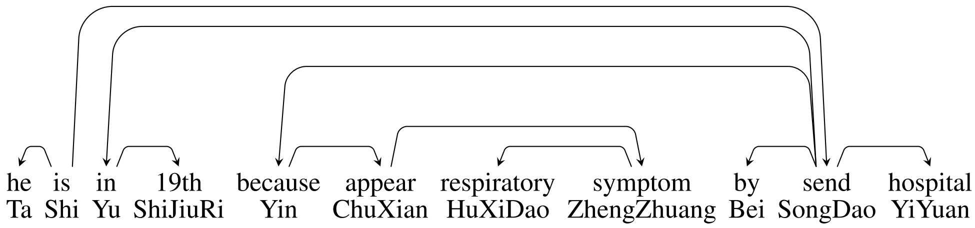
Table 5.2: Evaluation results of SERG and SNRG compared with HPBMT. $*$ means a system is significantly better than HPBMT at $p \leq 0.01$.

Chapter 4. However, instead of consolidating all experimental results to a single table, we present a sequence of result tables so that we can clearly explain various investigated aspects one by one.

5.4.1 Basic Results

We first compare SERG and SNRG with HPBMT. Table 5.2 shows evaluation results. We found that both SERG and SNRG achieve better BLEU (+1.4/+0.3 in terms of SNRG) and TER scores (-3.1/-0.3 in terms of SNRG) than HPBMT on both language pairs. We assume the improvement is mainly because dependency trees provide long-distance relations and help long-distance reordering (Xie et al., 2011). In terms of METEOR, while both our systems perform worse than HPBMT on ZH–EN (-0.5 in terms of SNRG), they achieve significant improvement on DE–EN (+0.2 in terms of SNRG). We found that on ZH–EN our systems generated shorter translations which may cause lower METEOR scores (He and Way, 2010).

Figure 5.13 shows translations of a Chinese sentence from the three systems. We found that both SERG and SNRG generated better translations than that of HPBMT. By tracking translation rules used during decoding, we found our systems used two rules to perform phrase reordering. Figure 5.14 shows a phrase alignment produced by SERG and SNRG. The phrase alignment is obtained by applying two reordering rules which helped our systems



Ref: he was sent to the hospital for respiratory symptoms on the 19th

HPBMT: he is in 19 due to respiratory symptoms were sent to the hospital

SERG: he was sent to hospital for respiratory symptoms on 19 september

SNRG: he was sent to hospital for respiratory symptoms on 19 september

Figure 5.13: Examples of translations from HPBMT, SERG, and SNRG. Compared with HPBMT, SERG and SNRG generated translations which have a better word order.

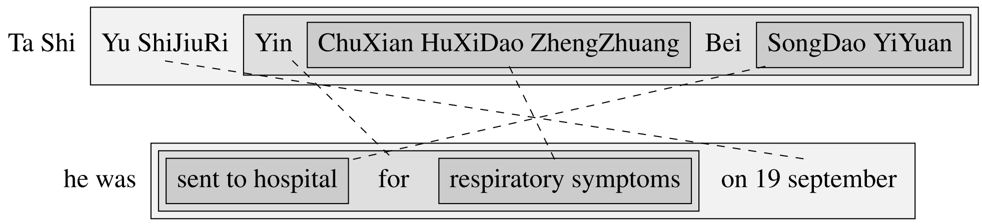


Figure 5.14: Phrase correspondence produced by SERG and SNRG for translating the example in Figure 5.13. Dashed lines indicate alignments between source phrases and target phrases.

perform better:

$$X \rightarrow \langle \text{Yin } X_{[1]} \text{ Bei } X_{[2]}, X_{[2]} \text{ for } X_{[1]} \rangle$$

$$X \rightarrow \langle \text{Yu ShiJiuRi } X_{[1]}, X_{[1]} \text{ on 19 september} \rangle$$

Note that for simplicity the two rules are represented in the form of HPB rules and graph edges are ignored.

5.4.2 Influence of Non-terminal Labels

Compared with HPBMT, our systems have an advantage of using linguistic non-terminals to help select rules (Zollmann and Venugopal, 2006; Almaghout et al., 2011; Zollmann and Vogel, 2011; Almaghout et al., 2012; Li et al., 2012a). Therefore, it would be interesting

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	SERG	37.7*	35.8*	20.6*	23.2*
	-NT	37.0	34.9	20.1	22.8
	SNRG	37.7*	35.8*	20.7	23.4
	-NT	37.2	34.7	20.7	23.6*
METEOR ↑	SERG	32.8	33.0	28.7*	30.0*
	-NT	33.1*	33.2*	28.2	29.4
	SNRG	32.6	32.7	28.7	30.0
	-NT	33.0*	33.0*	28.6	30.0
TER ↓	SERG	56.7*	57.9*	62.7*	59.1*
	-NT	58.2	59.5	63.2	59.6
	SNRG	56.2*	57.3*	62.5	59.1
	-NT	58.4	60.2	62.5	58.8*

Table 5.3: Evaluation results of SERG and SNRG without linguistic non-terminals (-NT). $*$ means a system is significantly better than its counterpart at $p \leq 0.01$.

to know translation performance of our systems when linguistic non-terminals are disabled. Table 5.3 shows evaluation results. We found that linguistic non-terminals significantly improve SERG in terms of BLEU (+0.8/+0.5) and TER (-1.6/-0.5) on both ZH–EN and DE–EN. In addition, SERG with linguistic non-terminals also achieves significantly better METEOR scores (+0.6) on DE–EN. While SNRG is less impacted by linguistic non-terminals on DE–EN, it is significantly improved by them on ZH–EN in terms of BLEU and TER (+0.8 BLEU, -2.6 TER).

Figure 5.15 shows examples of translations from SERG where linguistic non-terminals helped generate better translations. We found SERG used a translation rule:

$$\langle VV \rightarrow NR_{[1]} \text{ Ji Xu } NN\_PU_{[2]}, X \rightarrow X_{[1]} \text{ is urgently in need of } X_{[2]} \rangle$$

which correctly translated a Chinese phrase *sb Ji Xu sth* into English *sb is urgently in need of sth*. By contrast, SERG-NT chose the following rule:

$$\langle X \rightarrow X_{[1]} \text{ Ji Xu } X_{[2]}, X \rightarrow \text{ urgently needed } X_{[2]} X_{[1]} \rangle$$

which provided a worse translation option. Note that graph edges in rules are ignored.

NR	AD	VV	NN	PU
africa	urgent	need	assistance	.
FeiZhou	Ji	Xu	YuanZhu	.

Ref: africa is in urgent need of assistance .

SERG: africa is urgently in need of assistance .

SERG-NT: urgently needed aid to africa .

Figure 5.15: Examples of translations from SERG without linguistic non-terminals (-NT). Linguistic non-terminals helped to generate a better translation.

5.4.3 Influence of Span Limitation

In both HPBMT and our models, there is a limitation on the span length (also called maximum phrase length) which rules can be applied to during decoding. When a source span is larger than the limitation, glue rules are applied. This usually brings two benefits: (i) the decoding process becomes faster; (ii) because of the generalization capability of rules, which typically are learned under a length limitation, using them to translate very large spans may cause translation quality to deteriorate.

Therefore, we set this maximum phrase length to different values during decoding, including 10, 20 (default), 30, 40 and 50, to examine the generalization capability of rules in our systems. Figure 5.16 shows BLEU scores on all test sets. We found that on all different values, our systems achieve higher BLEU scores than HPBMT. Specifically, on ZH–EN HPBMT achieves its best performance when the maximum phrase length is 10 while our systems perform best at 20. This may suggest that our rules are more suitable to translate long phrases and thus have a better generalization capability than HPBMT rules. However, although on DE–EN our systems are more stable than HPBMT, all of them achieve the best performance at the same value of the parameter (i.e. 40). We presume this is because less long-distance reordering is needed on DE–EN than ZH–EN (Section 2.4).

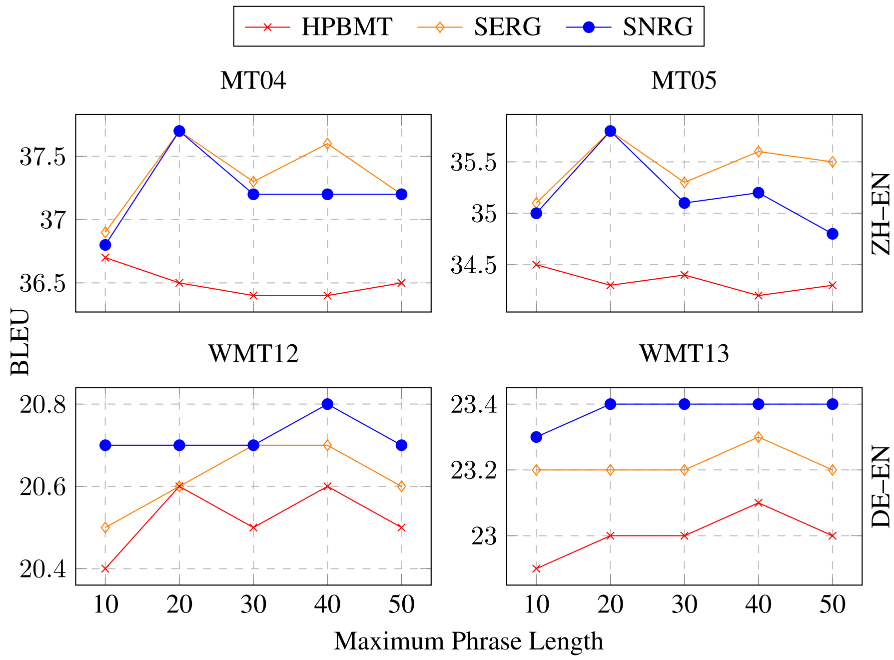


Figure 5.16: BLEU scores of HPBMT, SERG, and SNRG on all four test sets when the maximum phrase length during decoding is set to different values.

5.4.4 Influence of Sibling Links

Since our models take sibling relations into consideration, it would be interesting to know how these relations influence our systems. Since DSGs explicitly represent sibling links, it is trivial to conduct such experiments by directly removing sibling links (-Sib). Table 5.4 shows evaluation results. We found that on both ZH–EN and DE–EN, sibling links improve SNRG in terms of all three metrics (+3.8/+0.3 BLEU, +0.6/+0.2 METEOR, -4.0/-0.3 TER). The main reason for such improvements is that sibling links allow our model to extract much more rules (as shown in Table 5.5) which result in a higher phrase coverage. In addition, during decoding, sibling links help the decoder cover more non-syntactic phrases which are connected in graphs.

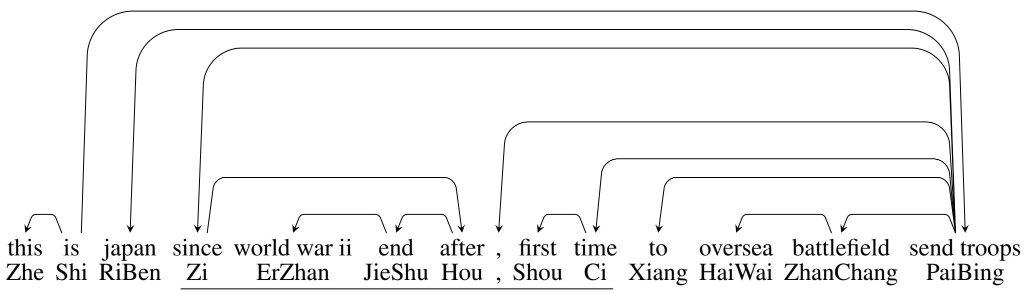
Figure 5.17 shows examples of translations. We found that the SNRG system generated a better translation for a Chinese phrase *Zi ErZhan JieShu Hou, Shou Ci*. The source phrase is not connected in trees but connected by sibling links in our graphs. By using rules which cover the phrase, SNRG performed a correct reordering on target words.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	SNRG	37.7*	35.8*	20.7*	23.4
	-Sib	33.7	32.0	20.4	23.2
METEOR ↑	SNRG	32.6*	32.7*	28.7*	30.0*
	-Sib	32.0	32.1	28.5	29.9
TER ↓	SNRG	56.2*	57.3*	62.5*	59.1
	-Sib	60.3	61.1	62.9	59.2

Table 5.4: Evaluation results of SNRG without sibling links (-Sib). $*$ means a system is significantly better than its counterpart at $p \leq 0.01$.

System	# Rules	
	ZH–EN	DE–EN
SNRG	382M	563M
-Sib	186M	364M

Table 5.5: The number of rules in SNRG without sibling links (-Sib).



Ref: this is the first time since the end of world war ii that japan has sent troops to overseas battlefields

SNRG: this is the first time since the end of world war ii on sending troops overseas battlefield

SNRG-Sib: this is in japan since the end of world war ii , for the first time to battlefield. overseas troops

Figure 5.17: Examples of translations from SNRG without sibling links (-Sib). With the help of sibling links, SNRG generated a better translation for the underlined source phrase which is not connected in the dependency tree.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	SNRG	37.7	35.8*	20.7	23.4
	+ET	37.6	35.4	20.8	23.5
METEOR ↑	SNRG	32.6	32.7	28.7	30.0
	+ET	32.9*	32.8*	28.7	30.0
TER ↓	SNRG	56.2*	57.3*	62.5	59.1
	+ET	56.8	58.1	62.4*	59.0

Table 5.6: Evaluation results of SNRG when edge types are considered (+ET). $*$ means a system is significantly better than its counterpart at $p \leq 0.01$.

System	# Rules	
	ZH–EN	DE–EN
SNRG	382M	563M
+ET	388M	580M

Table 5.7: The number of rules in SNRG with edge types (+ET).

5.4.5 Influence of Edge Types

In our model, DSGs combine dependency links and sibling links without distinction. In this section, we conduct experiments to examine the impact of taking edge types into consideration (+ET). Table 5.6 shows evaluation results. We found that overall edge types have no significant impact on translation performance of our system. This is reasonable since we found adding link types to rules did not significantly increase the number of rules in our systems, as shown in Table 5.7. This may suggest that when a rule is matched with a graph fragment in SNRG, in most cases edge types are matched as well.

5.4.6 Variance of Dependency Configurations

In Section 4.4.8, we compared D2S+Decomp with SegGBMT in terms of the variance of dependency configurations. We found that SegGBMT can cover more configurations than D2S+Decomp. In this section, we further compare the two systems in previous chapters with SERG and SNRG. Figure 5.18 lists examples of configurations. Compared with D2S+Decomp which covers sub-subtrees (Figure 5.18b) and subtrees (Figure 5.18d), we found that SERG and SNRG cover two more configurations: treelets (Figure 5.18e) and

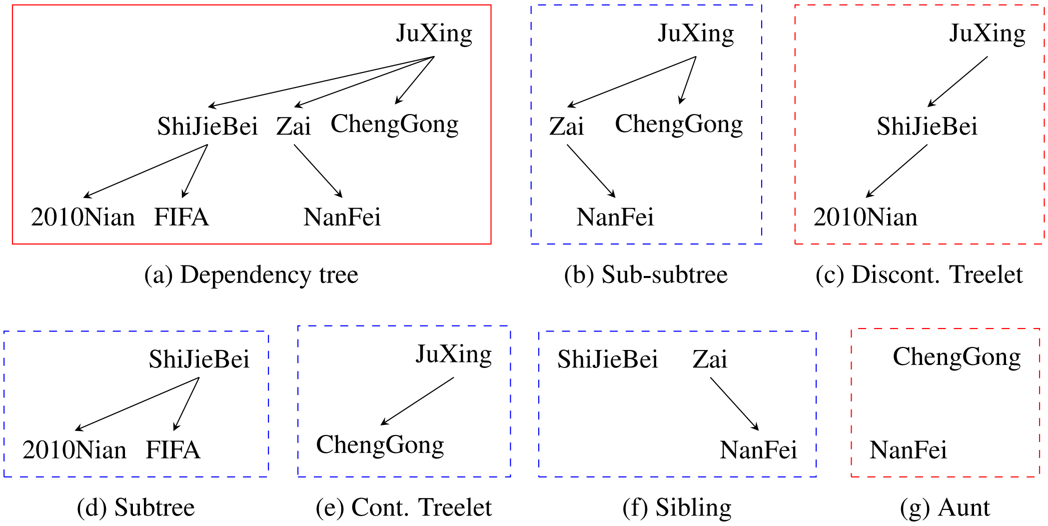


Figure 5.18: Given the dependency tree in (a), SERG and SNRG can cover dependency configurations (b), (d), (e), and (f). *Discontinuous (Discont.) Treelet* denotes a treelet covering a discontinuous phrase, while *Continuous (Cont.) Treelet* means a treelet covering a continuous phrase.

siblings (Figure 5.18f). However, compared with SegGBMT, SERG and SNRG cover fewer configurations. For example, SERG and SNRG cannot handle the aunt configuration in Figure 5.18g because it is not connected in DEGs and DSGs. In addition, SERG and SNRG can only translate treelets which cover continuous phrases, as in Figure 5.18e. Accordingly, the treelet in Figure 5.18c, which covers a discontinuous phrase, can only be translated by SegGBMT.

Table 5.8 shows evaluation results of different systems. We found that, compared with D2S+Decomp, SERG and SNRG are significantly better. Specifically, in terms of SNRG, the improvements are +1.0/0.5 (2.8%/2.3%, relative) BLEU, +0.6/0.3 (1.7%/1.0%, relative) METEOR, and -1.4/-0.9 (2.4%/1.5%, relative) TER on average on ZH–EN and DE–EN. This is reasonable since SERG and SNRG cover more phrases than D2S+Decomp. After integrating phrasal rules to D2S+Decomp, we found that the resulting system is comparable with SERG and SNRG. Compared with SegGBMT, even though SERG and SNRG have a lower coverage on dependency configurations, they achieve significantly better translation performance. Specifically, in terms of SNRG, the improvements are +3.2/+0.6 (9.5%/2.6%,

Metric	System	ZH–EN MT04	ZH–EN MT05	DE–EN WMT12	DE–EN WMT13
BLEU ↑	D2S+Decomp	36.6	34.9	20.4	22.7
	+Phrase	37.7	35.5	20.8	23.4
	SegGBMT	34.7	32.4	20.1	22.9
	SERG	$37.7^{*\ddagger}$	$35.8^{*\ddagger}$	$20.6^{\ddagger}$	$23.2^{*\ddagger}$
	SNRG	$37.7^{*\ddagger}$	$35.8^{*\ddagger}$	$20.7^{*\ddagger}$	$23.4^{*\ddagger}$
METEOR ↑	D2S+Decomp	32.1	32.1	28.4	29.7
	+Phrase	32.8	32.8	28.7	30.0
	SegGBMT	32.4	32.4	28.4	29.7
	SERG	$32.8^{*\ddagger}$	$33.0^{*+\ddagger}$	$28.7^{*\ddagger}$	$30.0^{*\ddagger}$
	SNRG	$32.6^{*\ddagger}$	$32.7^{*\ddagger}$	$28.7^{*\ddagger}$	$30.0^{*\ddagger}$
TER ↓	D2S+Decomp	57.5	58.7	63.4	60.0
	+Phrase	56.9	58.0	62.7	59.1
	SegGBMT	60.1	61.6	63.1	59.3
	SERG	$56.7^{*\ddagger}$	$57.9^{*\ddagger}$	$62.7^{*\ddagger}$	$59.1^{*\ddagger}$
	SNRG	$56.2^{*+\ddagger}$	$57.3^{*+\ddagger}$	$62.5^{*+\ddagger}$	$59.1^{*\ddagger}$

Table 5.8: Evaluation results of SERG and SNRG compared with D2S+Decomp in Chapter 3 and SegGBMT in Chapter 4. Note that +Phrase is an extra resource as described in Section 3.4.3. $*$, $+$, and $\ddagger$ mean SERG (or SNRG) is significantly better than D2S+Decomp, D2S+Decomp+Phrase, and SegGBMT at $p \leq 0.01$, respectively.

relative) BLEU, +0.3/+0.3 (0.7%/1.0%, relative) METEOR, and -3.6/-0.4 (5.7%/0.7%, relative) TER on average on ZH–EN and DE–EN. The main reason is that SERG and SNRG have a stronger capability to perform phrase reordering than SegGBMT.

5.5 Summary

In this chapter, we presented two translation models which are based on context-free graph grammars. To allow non-syntactic phrases to be covered, we construct graphs to implicitly or explicitly combine dependency links and sibling links. Experiments show that our models are significantly better than HPBMT. In addition, sibling links have a significant impact on translation performance of our systems.

However, rules in our current systems only cover continuous spans. Therefore, similar to conventional tree-based model, the generalization capability of our models is still confined by hard hierarchical constraints (Galley and Manning, 2010). In the next chapter, we will conclude and present avenues for future research.

Learn from yesterday, live for today, hope for tomorrow.

The important thing is not to stop questioning.

— Albert Einstein

Chapter 6

Conclusion

In this thesis, we have explored ways of building dependency graphs which combine source dependency trees with sequential information. We proposed translation models to translate these dependency graphs into target strings. By using graphs, our models combine the merits of both sequence-based and tree-based models and alleviate their weaknesses.

We started this thesis in Chapter 2 by reviewing previous translation models. According to the structures used in these models, we divided them into three categories: sequence-based, tree-based, and graph-based models. In each category, we introduced how these models are defined and translate sentences. We also revealed the shortcomings of existing models. In addition, we described experimental settings and baseline set-up throughout the thesis.

To better understand the motivation of using graphs, in Chapter 3 we presented a pseudo forest-to-string model which improves translation performance compared to a Dep2Str model by decomposing dependency structures into more fine-grained sub-structures. This decomposition allows the tree-based model to cover more phrases which are siblings. In experiments, we also showed that non-syntactic phrases connected by bigram relations (i.e. phrasal rules) have a significant impact on translation performance.

Based on the hypothesis that graphs are better representations of sentences for SMT, in Chapter 4 we proposed a segmentational graph-based translation model which translates a source graph by segmenting it into subgraphs. To make use of the merits of both sequences and trees, we constructed graphs which combine dependency relations and bigram relations. This makes our model a generalization of both the phrase-based model (Koehn et al., 2003) and dependency treelet-based model (Menezes and Quirk, 2005; Quirk et al., 2005). Experiments showed that our model significantly outperformed both.

Despite the advantages of the segmentational model in Chapter 4, it has a significant weakness in terms of phrase reordering. Therefore, inspired by using tree grammars to learn recursive translation rules, in Chapter 5 we proposed novel recursive graph-based translation models which are based on graph grammars. To alleviate the weakness of tree-based models regarding the translations of non-syntactic phrases, we constructed graphs which consider dependency relations and sibling relations. Experiments showed that our model is significantly better than the HPB model (Chiang, 2005; Chiang, 2007) and models in previous chapters.

Let us revisit our research questions proposed in Chapter 1:

RQ1 *Can we improve dependency tree-to-string translation models by incorporating more translation units implied by sequential relations?*

RQ2 *Can we construct dependency graphs which combine dependency relations and sequential relations in a unified representation?*

RQ3 *Can we translate input graphs into target sentences by graph segmentation where subgraphs, which may cover discontinuous phrases, are the basic translation units?*

RQ4 *Can we translate graphs by using graph grammars which parse the graphs and simultaneously generate target sentences using recursive translation rules?*

Chapter 3 provided a positive answer to **RQ1**. We showed that the proposed decomposition approach allows the Dep2Str model to cover more non-syntactic phrases which consist of siblings of a node in the tree structures. We also integrated phrasal rules which are directly introduced by bigram relations into the model. These phrasal rules further improved our model.

In Chapters 4 and 5, we constructed three kinds of dependency graphs in response to **RQ2**, including DBGs, DEGs, and DSGs. While DBGs combine dependency relations and bigram relations, DEGs and DSGs consist of dependency relations and sibling relations.

We tackled **RQ3** in Chapter 4 by presenting a segmentational graph-based translation model. The model segments DBGs into subgraphs which can then be translated into target phrases. A complete translation is generated by combining subgraph translations using beam search. In addition, we further improved this model using a graph segmentation model which helps to select a better subgraph to translate.

Recursive graph-based translation models which are based on context-free graph grammars were presented in Chapter 5 as a response to **RQ4**. The two graph grammars are node-replacement grammar and edge-replacement grammar, which translate node-labeled graphs (i.e. DSGs) and edge-labeled graphs (i.e. DEGs), respectively.

6.1 Contributions

In short, we have investigated ways of constructing dependency graphs and proposed translation models to translate these graphs. The contributions of this thesis are summarized as follows:

- **Pseudo forest-to-string translation model**. We presented a pseudo forest-to-string translation model which improves the Dep2Str model by dependency decomposition. We also presented a method to transform a dependency tree into a constituent-style tree which preserves dependency information. The transformation makes both our model and the Dep2Str model easier to implement in Moses. Experiments on ZH–EN and DE–EN demonstrated that our model achieves significantly better translation results.

- **Graph-construction methods**. We also presented three ways of constructing dependency graphs. These graphs combine dependency relations and sequential relations, including bigram relations and sibling relations. While dependency relations have the potential to provide long-distance links between words, sequential relations introduce local word-order into our graphs. By combining them together, our graphs contain richer information and provide more flexible translation units (i.e. subgraphs) than both phrases and subtrees.

- **Graph-based translation models**. To translate dependency graphs, we presented three graph-based translation models. A segmentational graph-based model translates a graph by segmenting it into subgraphs. The model is a generalization of the phrase-based model and treelet-based model. Two recursive graph-based models translate a graph by parsing the graph using graph grammars and simultaneously generating translations. While the non-recursive model has an advantage of covering more discontinuous phrases, recursive models have a stronger capability of performing phrase reordering.

6.2 Future Work

Using graphs in the field of NLP is becoming more popular. In recent years, researchers have built semantic treebanks (Banarescu et al., 2013) which use hypergraphs to represent the meanings of sentences. Subsequently, parsing these hypergraphs (Chiang et al., 2013; Flanigan et al., 2014; Artzi et al., 2015; Wang et al., 2015) to help language understanding and processing has become quite a hot topic. By contrast, using graphs in SMT is an under-explored but meaningful research direction, as discussed in this thesis. The research presented in this thesis can be strengthened by exploring more graph structures and corresponding translation models, and by using graphs in other MT paradigms, such as neural MT (NMT).

In Chapter 4 and 5 we presented graph-based translation models which translate graphs but with certain limitations. While the model in Chapter 4 takes discontinuity into consideration, it is weak at phrase reordering. By contrast, the models in Chapter 5 allow phrase reordering, but translation rules only cover continuous source spans for reasons of efficiency. Therefore, a more sophisticated model could be built by considering both discontinuity and reordering. Such a new model would be based on graph grammars but without the restriction defined in Section 5.2.2.

In this thesis, we constructed graphs which combine dependency relations and sequential relations. These graphs are designed to make use of the merits of both sequence-based models and tree-based models. In future, we would like to consider using other kinds of graphs, such as graphs representing feature structures which have proven to be a powerful tool for modeling morpho-syntactic aspects of natural languages (Graham, 2011; Williams, 2014). These kinds of graphs integrate morphological information and may be useful for translating morphologically-rich languages.

Recent progress on deep learning shows a promising way to perform MT with less feature engineering effort. Different from the conventional SMT framework as explained in this thesis, NMT represents each word using a numerical vector. Then, it uses neural networks to read source sentences and generates target words one by one in a left-to-right manner. However, current NMT models usually focus on sequence-to-sequence translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). Therefore, in the future, it would be interesting to explore how graphs can be used in NMT and how they impact on its translation performance.

Bibliography

Hala Almaghout, Jie Jiang, and Andy Way (2011). CCG Contextual Labels in Hierarchical Phrase-Based SMT. In: *Proceedings of The 15th Annual Conference of the European Association for Machine Translation*. Leuven, Belgium, pages 281–288.

Hala Almaghout, Jie Jiang, and Andy Way (2012). CCG-Based Syntactic Constraints in Hierarchical Phrase-Based SMT. In: *Proceedings of the 16th Annual Meeting of the European Association for Machine Translation*. Trento, Italy, pages 193–200.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer (2015). Broad-coverage CCG Semantic Parsing with AMR. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1699–1710.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2015). Neural Machine Translation By Jointly Learning to Align and Translate. In: *3th International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider (2013). Abstract Meaning Representation for Sembanking. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria, pages 178–186.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra (1996). A Maximum Entropy Approach to Natural Language Processing. In: *Computational Linguistics* 22.1, pages 39–71.

Bernd Bohnet (2010). Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Volume 2)*. Beijing, China, pages 89–97.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul Rossin (1990). A Statistical Approach to Machine Translation. In: *Computational Linguistics* 16.2, pages 76–85.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, Robert L. Mercer, and Paul Roossin (1988). A Statistical Approach to Language Translation. In: *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*. Budapest, Hungry, pages 71–76.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. In: *Computational Linguistics* 19.2, pages 263–311.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning (2008). Optimizing Chinese Word Segmentation for Machine Translation Performance. In: *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio, USA, pages 224–232.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning (2009). Discriminative Reordering with Chinese Grammatical Relations Features. In: *Proceedings of SSST-3, Third Workshop on Syntax and Structure in Statistical Translation*. Boulder, Colorado, USA, pages 51–59.

Hongshen Chen, Jun Xie, Fandong Meng, Wenbin Jiang, and Qun Liu (2014). A Dependency Edge-Based Transfer Model for Statistical Machine Translation. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland, pages 1103–1113.

Stanley F. Chen and Joshua Goodman (1996). An Empirical Study of Smoothing Techniques for Language Modeling. In: *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. Santa Cruz, California, pages 310–318.

Colin Cherry (2013). Improved Reordering for Phrase-Based Translation Using Sparse Features. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, USA, pages 22–31.

Colin Cherry and George Foster (2012). Batch Tuning Strategies for Statistical Machine Translation. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montreal, Canada, pages 427–436.

David Chiang (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Ann Arbor, Michigan, USA, pages 263–270.

David Chiang (2007). Hierarchical Phrase-Based Translation. In: *Computational Linguistics* 33.2, pages 201–228.

David Chiang (2012). Grammars for Language and Genes: Theoretical and Empirical Investigations. Springer.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight (2013). Parsing Graphs with Hyperedge Replacement Grammars. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 924–932.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1724–1734.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith (2011). Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:*

Human Language Technologies (Volume 2: Short Papers). Portland, Oregon, pages 176–181.

John Cocke and Jacob T. Schwartz (1970). Programming Languages and Their Compilers: Preliminary Notes. Tech. rep. New York, NY: Courant Institute of Mathematical Sciences, New York University.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann (1999). A Statistical Parser for Czech. In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*. College Park, Maryland, pages 505–512.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society, Series B* 39.1, pages 1–38.

Michael Denkowski and Alon Lavie (2011). Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, pages 85–91.

Bonnie J. Dorr (1994). Machine Translation Divergences: A Formal Description and Proposed Solution. In: *Computational Linguistics* 20.4, pages 597–633.

Frank Drewes, Hans Jörg Kreowski, and Annegret Habel (1997). Hyperedge Replacement Graph Grammars. In: *Handbook of Graph Grammars and Computing by Graph Transformation*. Ed. by Grzegorz Rozenberg. River Edge, NJ, USA: World Scientific Publishing Co., Inc., pages 95–162.

Jason Eisner (2003). Learning Non-isomorphic Tree Mappings for Machine Translation. In: *the Companion Volume to Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Sapporo, Japan, pages 205–208.

Eva M. Duran Eppler (2013). Dependency Distance and Bilingual Language Use: Evidence from German/English and Chinese/English Data. In: *Proceedings of the Second International Conference on Dependency Linguistics*. Prague, Czech Republic, pages 78–87.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith (2014). A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pages 1426–1436.

Heidi J. Fox (2002). Phrasal Cohesion and Statistical Machine Translation. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Philadelphia, USA, pages 304–3111.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer (2006). Scalable Inference and Training of Context-Rich Syntactic Translation Models. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 961–968.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu (2004). What's in a Translation Rule? In: *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Boston, Massachusetts, USA, pages 273–280.

Michel Galley and Christopher D. Manning (2008). A Simple and Effective Hierarchical Phrase Reordering Model. In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii, USA, pages 848–856.

Michel Galley and Christopher D. Manning (2010). Accurate Non-hierarchical Phrase-Based Translation. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California, USA, pages 966–974.

Yvette Graham (2011). Deep Syntax in Statistical Machine Translation. PhD thesis. Dublin City University.

Greg Hanneman and Alon Lavie (2009). Decoding with Syntactic and Non-syntactic Phrases in a Syntax-Based Machine Translation System. In: *Proceedings of SSST-3, Third Work-*

shop on Syntax and Structure in Statistical Translation. Boulder, Colorado, USA, pages 1–9.

Yifan He and Andy Way (2010). Metric and Reference Factors in Minimum Error Rate Training. In: *Machine Translation* 24.1, pages 27–38.

Zhongjun He, Qun Liu, and Shouxun Lin (2008). Improving Statistical Machine Translation Using Lexicalized Rule Selection. In: *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*. Manchester, United Kingdom, pages 321–328.

Hieu Hoang (2011). Improving Statistical Machine Translation with Linguistic Information. PhD thesis. University of Edinburgh.

Liang Huang, Kevin Knight, and Aravind Joshi (2006a). A Syntax-Directed Translator with Extended Domain of Locality. In: *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*. New York City, New York, pages 1–8.

Liang Huang, Kevin Knight, and Aravind Joshi (2006b). Statistical Syntax-Directed Translation with Extended Domain of Locality. In: *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*. Cambridge, Massachusetts, USA, pages 66–73.

Liang Huang and Haitao Mi (2010). Efficient Incremental Decoding for Tree-to-String Translation. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA, pages 273–283.

Matthias Huck, Hieu Hoang, and Philipp Koehn (2014). Augmenting String-to-Tree and Tree-to-String Translation with Non-Syntactic Phrases. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA, pages 486–498.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight (2012). Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In: *Proceedings of COLING 2012, the 24th International Conference on Computational Linguistics: Technical Papers*. Mumbai, India, pages 1359–1376.

Rasoul Samed Zadeh Kaljahi, Raphael Rubino, Johann Roturier, and Jennifer Foster (2012). A Detailed Analysis of Phrase-Based and Syntax-Based Machine Translation: The Search for Systematic Differences. In: *Proceedings The Tenth Biennial Conference of the Association for Machine Translation in the Americas*. San Diego, CA, pages 1103–1113.

Tadao Kasami (1965). An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. Tech. rep. Bedford, MA: Air Force Cambridge Research Lab.

Kevin Knight (1999). Decoding Complexity in Word-replacement Translation Models. In: *Computational Linguistics* 25.4, pages 607–615.

Philipp Koehn (2010). Statistical Machine Translation. 1st. New York, NY, USA: Cambridge University Press.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot (2005). Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In: *Proceedings of the International Workshop on Spoken Language Translation 2005*. Pittsburgh, PA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu (2003). Statistical Phrase-Based Translation. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Edmonton, Canada, pages 48–54.

Jacek Kukluk (2007). Inference of Node and Edge Replacement Graph Grammars. PhD thesis. University of Texas at Arlington.

Jacek P. Kukluk, Lawrence B. Holder, and Diane J. Cook (2008). Inference of edge replacement graph grammars. In: *International Journal on Artificial Intelligence Tools* 17.3, pages 539–554.

Clemens Lautemann (1990). The Complexity of Graph Languages Generated by Hyperedge Replacement. In: *Acta Informatica* 27.5, pages 399–421.

P.M. Lewis II and R.E. Stearns (1968). Syntax-Directed Transduction. In: *Journal of the ACM* 15.3, pages 465–488.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith (2012a). Head-Driven Hierarchical Phrase-Based Translation. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea, pages 33–37.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith (2012b). Using Syntactic Head Information in Hierarchical Phrase-Based Translation. In: *Proceedings of the 7th Workshop on Statistical Machine Translation*. Montréal, Canada, pages 232–242.

Dekang Lin (2004). A Path-Based Transfer Model for Machine Translation. In: *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland, pages 625–630.

Lemao Liu and Liang Huang (2014). Search-Aware Tuning for Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1942–1952.

Yang Liu and Qun Liu (2010). Joint Parsing and Translation. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Volume 2)*. Beijing, China, pages 707–715.

Yang Liu, Qun Liu, and Shouxun Lin (2006). Tree-to-String Alignment Template for Statistical Machine Translation. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 609–616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight (2006). SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia, pages 44–52.

Arul Menezes and Chris Quirk (2005). Dependency Treelet Translation: The Convergence of Statistical and Example-Based Machine-Translation? In: *Proceedings of the Workshop on Example-Based Machine Translation*. Phuket, Thailand.

Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü, and Qun Liu (2013). Translation with Source Constituency and Dependency Trees. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pages 1066–1076.

Haitao Mi, Liang Huang, and Qun Liu (2008). Forest-Based Translation. In: *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Columbus, Ohio, USA, pages 192–199.

Rebecca Nesson, Stuart M. Shieber, and Alexander Rush (2006). Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation. In: *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*. Cambridge, Massachusetts, USA, pages 128–137.

Joakim Nivre and Jens Nilsson (2005). Pseudo-Projective Dependency Parsing. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Ann Arbor, Michigan, USA, pages 99–106.

Franz Josef Och (1999). An Efficient Method for Determining Bilingual Word Classes. In: *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*. Bergen, Norway, pages 71–76.

Franz Josef Och and Hermann Ney (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 295–302.

Franz Josef Och and Hermann Ney (2003). A Systematic Comparison of Various Statistical Alignment Models. In: *Computational Linguistics* 29.1, pages 19–51.

Franz Josef Och and Hermann Ney (2004). The Alignment Template Approach to Statistical Machine Translation. In: *Computational Linguistics* 30.4, pages 417–449.

Franz Josef Och, Christoph Tillmann, and Hermann Ney (1999). Improved Alignment Models for Statistical Machine Translation. In: *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 311–318.

Chris Quirk and Simon Corston-Oliver (2006). The Impact of Parse Quality on Syntactically-informed Statistical Machine Translation. In: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia, pages 62–69.

Chris Quirk, Arul Menezes, and Colin Cherry (2005). Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Ann Arbor, Michigan, USA, pages 271–279.

Grzegorz Rozenberg, ed. (1997). Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations. River Edge, NJ, USA: World Scientific Publishing Co., Inc.

Libin Shen, Jinxi Xu, and Ralph Weischedel (2010). String-to-Dependency Statistical Machine Translation. In: *Computational Linguistics* 36.4, pages 649–671.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul (2006). A Study of Translation Edit Rate with Targeted Human Annotation. In: *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*. Cambridge, Massachusetts, USA, pages 223–231.

Andreas Stolcke (2002). SRILM – An Extensible Language Modeling Toolkit. In: *Proceedings of the 7th International Conference on Spoken Language Processing*. Denver, Colorado, USA, pages 901–904.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le (2014). Sequence to Sequence Learning with Neural Networks. In: *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin (2010). Dependency Forest for Statistical Machine Translation. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Volume 2)*. Beijing, China, pages 1092–1100.

Chuan Wang, Nianwen Xue, and Sameer Pradhan (2015). A Transition-Based Algorithm for AMR Parsing. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, pages 366–375.

Wei Wang, Kevin Knight, and Daniel Marcu (2007). Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Prague, Czech Republic, pages 746–754.

Philip Williams (2014). Unification-Based Constraints for Statistical Machine Translation. PhD thesis. University of Edinburgh.

Fei Xia and Martha Palmer (2001). Converting Dependency Structures to Phrase Structures. In: *Proceedings of the First International Conference on Human Language Technology Research*. San Diego, pages 1–5.

Jun Xie, Haitao Mi, and Qun Liu (2011). A Novel Dependency-to-String Model for Statistical Machine Translation. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, United Kingdom, pages 216–226.

Jun Xie, Jinan Xu, and Qun Liu (2014). Augment Dependency-to-String Translation with Fixed and Floating Structures. In: *Proceedings of COLING 2014, the 25th International*

Conference on Computational Linguistics: Technical Papers. Dublin, Ireland, pages 2217–2226.

Deyi Xiong, Qun Liu, and Shouxun Lin (2006). Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 521–528.

Deyi Xiong, Qun Liu, and Shouxun Lin (2007). A Dependency Treelet String Correspondence Model for Statistical Machine Translation. In: *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, pages 40–47.

Kenji Yamada and Kevin Knight (2001). A Syntax-Based Statistical Translation Model. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Toulouse, France, pages 523–530.

Kenji Yamada and Kevin Knight (2002). A Decoder for Syntax-Based Statistical MT. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 303–310.

Daniel H. Younger (1967). Recognition and Parsing of Context-Free Languages in Time n^3. In: *Information and Control* 10.2, pages 189–208.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight (2006). Synchronous Binarization for Machine Translation. In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. New York City, USA, pages 256–263.

Min Zhang, Hongfei Jiang, Aiti Aw, Sun Jun, Sheng Li, and Chew Lim Tan (2007). A Tree-to-Tree Alignment-Based Model for Statistical Machine Translation. In: *Proceedings of Machine Translation Summit XI*. Copenhagen, Denmark, pages 535–542.

Andreas Zollmann and Ashish Venugopal (2006). Syntax Augmented Machine Translation via Chart Parsing. In: *Proceedings of the Workshop on Statistical Machine Translation*. New York City, USA, pages 138–141.

Andreas Zollmann and Stephan Vogel (2011). A Word-Class Approach to Labeling PSCFG Rules for Machine Translation. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Portland, Oregon, pages 1–11.